

311-CD-108-001

**EOSDIS Core System Project**

**Storage Management Database Design  
and Schema Specifications  
for the ECS Project**

**This document has not yet been approved by the  
Government for general use or distribution.**

Draft

January 1998

Raytheon Systems Company  
Upper Marlboro, Maryland

# **Storage Management Database Design and Schema Specifications for the ECS Project**

**Draft**

**January 1998**

Prepared Under Contract NAS5-60000  
CDRL Item #050

## **RESPONSIBLE ENGINEER**

Mary S. Armstrong for /s/ 12/31/97  
Maureen Muganda Date  
EOSDIS Core System Project

## **SUBMITTED BY**

T. W. Fisher /s/ 12/31/97  
Terry Fisher, ECS CCB Chairman Date  
EOSDIS Core System Project

**Raytheon Systems Company**  
Upper Marlboro, Maryland

This page intentionally left blank.

# Preface

---

This document describes the data design and database specification for the Storage Management subsystem. It is one of ten documents comprising the detailed database design specifications for each of the ECS subsystems.

The subsystem database design specifications for the as delivered system include:

311-CD-100 Access Control (ACL) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-101 Data Distribution (DDIST) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-102 Data Management (DM) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-103 Ingest Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-104 Interoperability Subsystem (IOS) Database Design and Database Schema Specifications for the ECS Project

311-CD-105 Management Support Subsystem (MSS) Database Design and Database Schema Specifications for the ECS Project

311-CD-106 Planning and Data Processing Subsystem (PDPS) Database Design and Database Schema Specifications for the ECS Project

311-CD-107 Science Data Server (SDSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-108 Storage Management (STMGMT) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-109 Subscription Server (SUBSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project

This submittal meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. It is a formal contract deliverable with an approval code 1. It requires Government review and approval prior to acceptance and use. This document is under ECS contractor configuration control. Once approved, contractor approved changes will be handled in accordance with Class I and class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by Document Change Notice (DCN) or by complete revision.

Entity Relationship Diagrams (ERDs) presented in this document have been exported directly from tools and some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these drawings on-line using the Portable Document Format (PDF) electronic copy available via the ECS Data Handling System (ECS) on the world-wide web at <http://edhs1.gsfc.nasa.gov>.

Any questions should be addressed to:

Data Management Office  
The ECS Project Office  
Raytheon Systems Company  
1616 McCormick Drive  
Upper Marlboro, Maryland 20774-5372

## **Abstract**

---

This document outlines “as-built” database design and database schema of the Storage Management database including the physical layout of the database and initial installation parameters.

**Keywords:** data, database, design, configuration, database installation, scripts, security, data model, data dictionary, replication, performance tuning, SQL server, database security, replication, database scripts

This page intentionally left blank.

# Change Information Page

---

List of Effective Pages	
Page Number	Issue
Title	Draft
iii through xii	Draft
1-1 and 1-2	Draft
2-1 and 2-2	Draft
3-1 and 3-2	Draft
4-1 through 4-186	Draft
5-1 and 5-2	Draft
6-1 and 6-2	Draft
7-1 and 7-2	Draft
8-1 and 8-2	Draft
AB-1 through AB-8	Draft

Document History			
Document Number	Status/Issue	Publication Date	CCR Number
311-CD-108-001	Draft	January 1998	97-1755

This page intentionally left blank.

# **Contents**

---

## **Preface**

## **Abstract**

### **1. Introduction**

1.1	Identification .....	1-1
1.2	Scope.....	1-1
1.3	Purpose.....	1-1
1.4	Audience .....	1-1

### **2. Related Documentation**

2.1	Parent Documents .....	2-1
2.2	Applicable Documents.....	2-1
2.3	Information Documents .....	2-1
2.3.1	Information Documents Referenced .....	2-1
2.3.2	Information Documents Not Referenced .....	2-1

### **3. Database Configurations**

3.1	Server Configurations .....	3-1
3.2	Storage Device Layouts .....	3-1

## **4. Data Design**

4.1	Database Overview .....	4-1
4.1.1	Physical Data Model Entity Relationship Diagram .....	4-1
4.1.2	Tables .....	4-5
4.1.3	Columns.....	4-17
4.1.4	Column Domains.....	4-36
4.1.5	Rules.....	4-48
4.1.6	Defaults .....	4-48
4.1.7	Views.....	4-48
4.1.8	Integrity Constraints.....	4-48
4.1.9	Triggers .....	4-51
4.1.10	Stored Procedures.....	4-59
4.2	File Usage .....	4-186
4.2.1	Files Definitions .....	4-186
4.2.2	Attributes.....	4-186
4.2.3	Attribute Domains .....	4-186

## **5. Performance and Tuning Factors**

5.1	Indexes .....	5-1
5.2	Segments .....	5-1
5.3	Named Caches.....	5-1

## **6. Database Security**

6.1	Initial Users .....	6-1
6.2	Groups.....	6-1
6.3	Roles.....	6-1
6.4	Object Permissions.....	6-2

## **7. Replication**

7.1	Replication Overview .....	7-1
7.2	Replication Definitions .....	7-1
7.3	Replication Subscriptions .....	7-1
7.4	Replication Database Configuration .....	7-1
7.5	Replication Server Configuration .....	7-1

## **8. Scripts 1**

8.1	Installation Scripts.....	8-1
8.2	De-Installation Scripts.....	8-1
8.3	Backup/Recovery Scripts.....	8-1
8.4	Miscellaneous Scripts .....	8-1

## **Abbreviations and Acronyms**

## **Figures**

4-1	Sample ERD.....	4-1
4-2	Storage Management ERD.....	4-3

This page intentionally left blank.

# **1. Introduction**

---

## **1.1 Identification**

This Storage Management (STMGMT) Database Design and Database Schema Specification document, Contract Data Requirement List (CDRL) Item Number 050, whose requirements are specified in Data Item description (DID\_ 311/DV1, is a required deliverable under the Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS), Contract NAS5-60000.

## **1.2 Scope**

The STMGMT Database Design and Database Schema Specification document describes the data design and database specifications to support the data requirements of Release 2 Drop 3 STMGMT software.

## **1.3 Purpose**

The purpose of the STMGMT Database Design and Database Schema Specification document is to support the maintenance of STMGMT data and databases throughout the life cycle of ECS. This document communicates the database implementation in sufficient detail to support ongoing configuration management.

## **1.4 Audience**

This document is intended to be used by ECS maintenance and operations staff. The document is organized as follows:

Section 1 provides information regarding the identification, purpose, scope and audience of this document.

Section 2 provides a listing of the related documents, which were used as a source of information for this document.

Section 3 provides a mapping data bases to hardware components.

Section 4 contains the STMGMT physical data model which is the database tables, triggers, stored procedures, and flat files.

Section 5. provides a description of database performance and tuning features such as indexes, caches, and data segments.

Section 6 provides a description of the security infrastructure used and list of the users, groups, and permissions available upon initial installation.

Section 7 contains replication design and implementation details.

Section 8 provides a description of database and database related scripts used for installation, de-installation, backup/recovery, and other miscellaneous functions.

## **2. Related Documentation**

---

### **2.1 Parent Documents**

The parent document is the document from which this document's scope and content are derived.

TBS

### **2.2 Applicable Documents**

The following documents, including Internet links, are referenced in the STMGMT Database Design and Database Schema Specification, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this volume. Internet links cannot be guaranteed for accuracy or currency.

305-CD-100-001	Design Specification Overview for the ECS Project
920-TDG-009	Release B0 GSFC DAAC Database Information
920-TDN- 009	Release B0 NSIDC DAAC Database Information
920-TDE-009	Release B0 EDC DAAC Database Information
920-TDL-009	Release B0 LARC DAAC Database Information
920-TDS-00X	Release B0 SMC DAAC Database Information
920-TDM-009	Release B0 Mini-DAAC Database Information

### **2.3 Information Documents**

#### **2.3.1 Information Documents Referenced**

The following documents are referenced herein and, amplify or clarify the information presented in this document. These documents are not binding on the content of this document.

TBS

#### **2.3.2 Information Documents Not Referenced**

The following documents, although not referenced herein and/or not directly applicable, do amplify or clarify the information presented in this document. These documents are not binding on the content of this document.

TBS

This page intentionally left blank.

### **3. Database Configurations**

---

#### **3.1 Server Configurations**

The database configuration of the STMGMT server varies from DAAC to DAAC based on individualized DAAC requirements and hardware availability. These DAAC-specific database configurations are detailed on the following documents:

920-TDG-009 Release B0 GSFC DAAC Database Information

920-TDN- 009 Release B0 NSIDC DAAC Database Information

920-TDE-009 Release B0 EDC DAAC Database Information

920-TDL-009 Release B0 LARC DAAC Database Information

920-TDS-009 Release B0 SMC DAAC Database Information

920-TDM-009 Release B0 Mini-DAAC Database Information

These documents are maintained as part of the ECS baseline and available on the world-wide web at the URL <http://pete.hitc.com/baseline/>.

#### **3.2 Storage Device Layouts**

Storage Device layouts, disk partitions, vary from DAAC to DAAC based on the amount of data storage expected to be needed to accommodate a particular DAAC's storage requirements. Disk partitions for the STMGMT server at each DAAC is detailed in the following documents:

922-TDG-XXX Release B0 STMGMT Disk Partitions

922-TDN-XXX Release B0 STMGMT Disk Partitions

922-TDE-XXX Release B0 STMGMT Disk Partitions

922-TDL-XXX Release B0 STMGMT Disk Partitions

922-TDS-XXX Release B0 STMGMT Disk Partitions

922-TDM-XXX Release B0 STMGMT Disk Partitions

These documents are maintained as part of the ECS baseline and available on the world-wide web at the URL <http://pete.hitc.com/baseline/>.

This page intentionally left blank.

## 4. Data Design

---

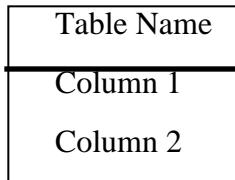
### 4.1 Database Overview

The STMGMT database implements the large majority of the persistent data requirements for the STMGMT subsystem. The database is designed in such a manner as to satisfy business policy while maintaining data integrity and consistency. Database tables are implemented using the Sybase Relational Database Management system (DBMS) version 11.0.1. All components of the STMGMT database are described in the section which follow in sufficient detail to support maintenance needs.

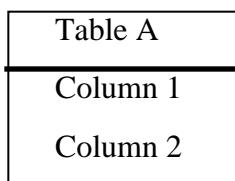
#### 4.1.1 Physical Data Model Entity Relationship Diagram

The Entity Relationship Diagram(ERD) presents a schematic depiction of the STMGMT physical data model. The ERDs presented here for the STMGMT database were produced using the S-Designor Data Architect Computer Aided Software Engineering (CASE) tool. ERDs represent the relationship between entities or database tables. On ERDs, tables are represented by rectangles and relationships are represented as arrow (see Figure 4-1).

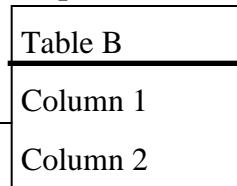
##### Sample Table



##### Independent Table



##### Dependent Table



*Table A has zero-to-many relationship with Table B*

**Figure 4-1. Sample ERD**

The ERDs for the STMGMT database are shown in Figures 4-2.

This page intentionally left blank.

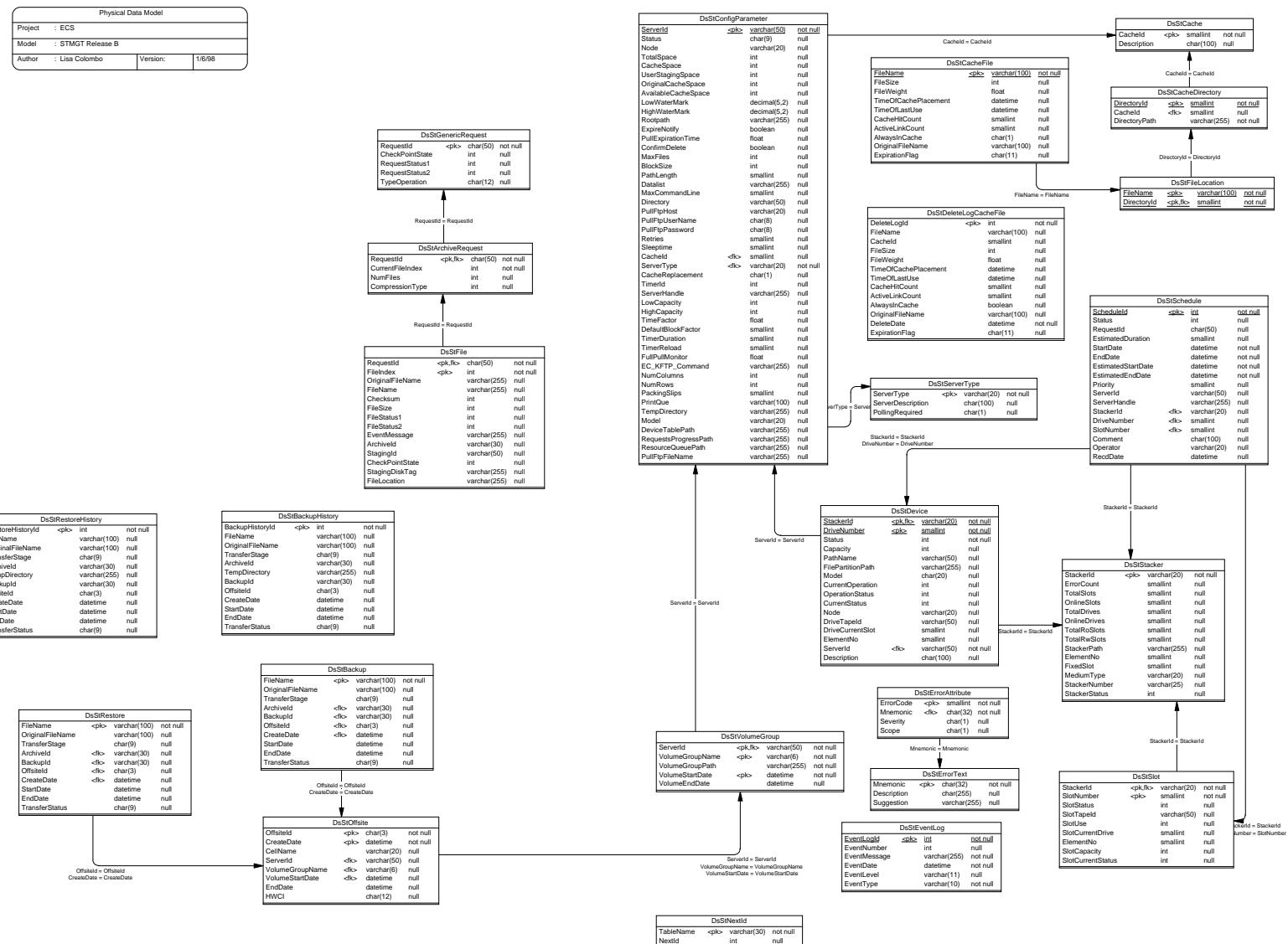


Figure 4-2. Storage Management ERD

#### **4.1.2 Tables**

A listing of each the tables in the STMGMT database is given here. A brief definition of each of these tables follows including a listing of the columns comprising the table. The Column List indicates if the column is part of the primary key for the table. That is if the columns can be used alone or in combination with other primary key columns to uniquely identify a single row in the table. The column list also indicates whether the column is a mandatory attribute that must be included in every row.

Table Code
Drives
DsStBackup
DsStBackupHistory
DsStCache
DsStCacheDirectory
DsStCacheFile
DsStConfigParameter
DsStDeleteLogCacheFile
DsStDevice
DsStErrorAttribute
DsStErrorText
DsStEventLog
DsStFileLocation
DsStNextId
DsStOffsite
DsStRestore
DsStRestoreHistory
DsStSchedule
DsStServerType
DsStSlot
DsStStacker
DsStVolumeGroup
pull_list
pull_link
Request
UniversalRef
Slots
Stackers

## Table: Drives

### Description

Originally implemented as the entity to contain an entry for each peripheral device that Storage Management uses to service requests to Ingest or Distribute data. Since this entity acts as a look-up entity for entries that exist in the Request entity, the data in this entity will be initialized prior to software installation.

Replaced by new table DsStDevice.

### Column List

Name	Code	Type	PK	Mandatory
drive_current_slot	drive_current_slot	int	No	No
drive_number	drive_number	int	Yes	Yes
drive_path	drive_path	varchar(50)	No	Yes
drive_status	drive_status	int	No	No
drive_tapeid	drive_tapeid	varchar(50)	No	No
element_no	element_no	int	No	No
stacker_id	stacker_id	varchar(50)	Yes	Yes

## Table: DsStBackup

### Description

Stores a reference to every file currently being backed-up related to Archive Backup and Restore functionality.

### Column List

Name	Code	Type	PK	Mandatory
Archiveld	Archiveld	varchar(30)	No	No
BackupId	BackupId	varchar(30)	No	No
EndDate	EndDate	datetime	No	No
FileName	FileName	varchar(100)	Yes	Yes
Offsiteld	Offsiteld	char(3)	No	No
CreateDate	CreateDate	datetime	No	No
OriginalFileName	OriginalFileName	varchar(100)	No	No
StartDate	StartDate	datetime	No	No
TransferStage	TransferStage	char(9)	No	No
TransferStatus	TransferStatus	char(9)	No	No

## **Table: DsStBackupHistory**

### **Description**

Stores a history of every file successfully backed-up related to Archive Backup and Restore functionality.

### **Column List**

Name	Code	Type	PK	Mandatory
Archiveld	Archiveld	varchar(30)	No	No
BackupHistoryId	BackupHistoryId	int	Yes	Yes
BackupId	BackupId	varchar(30)	No	No
EndDate	EndDate	datetime	No	No
FileName	FileName	varchar(100)	No	No
Offsiteld	Offsiteld	char(3)	No	No
CreateDate	CreateDate	datetime	No	No
OriginalFileName	OriginalFileName	varchar(100)	No	No
StartDate	StartDate	datetime	No	No
TempDirectory	TempDirectory	varchar(255)	No	No
TransferStage	TransferStage	char(9)	No	No
TransferStatus	TransferStatus	char(9)	No	No

## **Table: DsStCache**

### **Description**

Table to identify every instance of a cache related to Pull Monitor Cache Management. Currently defaulted to only 1 existing cache. Will be expanded when Centralized Cache Management functionality implemented.

### **Column List**

Name	Code	Type	PK	Mandatory
Cacheld	Cacheld	smallint	Yes	Yes
Description	Description	char(100)	No	No

## **Table: DsStCacheDirectory**

### **Description**

Contains an entry for each directory that comprises a Storage Management cache. An entry is inserted into the entity every time a new cache is created, or a directory is added to an existing cache.

## Column List

Name	Code	Type	PK	Mandatory
Cacheld	Cacheld	smallint	No	No
DirectoryId	DirectoryId	smallint	Yes	Yes
DirectoryPath	DirectoryPath	varchar(255)	No	Yes

## Table: DsStCacheFile

### Description

Contains an entry for each file that Storage Management Pull Monitor is currently processing (originally implemented via the pull\_list table). An entry is inserted into the entity for each file retrieved from the archive (AMASS). DsStFileLocation (originally implemented via the pull\_link table) will track the individual cache locations of (or links to) the file.

## Column List

Name	Code	Type	PK	Mandatory
ActiveLinkCount	ActiveLinkCount	smallint	No	No
AlwaysInCache	AlwaysInCache	char(1)	No	No
CacheHitCount	CacheHitCount	smallint	No	No
ExpirationFlag	ExpirationFlag	char(11)	No	No
FileName	FileName	varchar(100)	Yes	Yes
FileSize	FileSize	int	No	No
FileWeight	FileWeight	float	No	No
OriginalFileName	OriginalFileName	varchar(100)	No	No
TimeOfCachePlacement	TimeOfCachePlacement	datetime	No	No
TimeOfLastUse	TimeOfLastUse	datetime	No	No

## Table: DsStConfigParameter

### Description

Contains an entry for information necessary to configure and initialize each DsStServerType supported by Storage Management. The data will consist of information currently accessed through configuration files (\*.CFG) plus information as it pertains to the status and node of operation for each server. An entry is inserted for each parameter that a server uses.

Two types of parameters may be defined within the constructs of the DsStConfigParameter table, startup and runtime. Startup parameters require the associated server be restarted in order for the parameters to be used. Runtime parameters may be changed without restarting the server (i.e. the server periodically queries the configuration table for new values).

## Column List

Name	Code	Type	PK	Mandatory
AvailableCacheSpace	AvailableCacheSpace	int	No	No
BlockSize	BlockSize	int	No	No
Cacheld	Cacheld	smallint	No	No
CacheReplacement	CacheReplacement	char(1)	No	No
CacheSpace	CacheSpace	int	No	No
ConfirmDelete	ConfirmDelete	boolean	No	No
Datalist	Datalist	varchar(255)	No	No
DefaultBlockFactor	DefaultBlockFactor	smallint	No	No
DeviceTableName	DeviceTableName	varchar(255)	No	No
Directory	Directory	varchar(50)	No	No
EC_KFTP_Command	EC_KFTP_Command	varchar(255)	No	No
ExpireNotify	ExpireNotify	boolean	No	No
FullPullMonitor	FullPullMonitor	float	No	No
HighCapacity	HighCapacity	int	No	No
HighWaterMark	HighWaterMark	decimal(5,2)	No	No
LowCapacity	LowCapacity	int	No	No
LowWaterMark	LowWaterMark	decimal(5,2)	No	No
MaxCommandLine	MaxCommandLine	smallint	No	No
MaxFiles	MaxFiles	int	No	No
Model	Model	varchar(20)	No	No
Node	Node	varchar(20)	No	No
NumColumns	NumColumns	int	No	No
NumRows	NumRows	int	No	No
OriginalCacheSpace	OriginalCacheSpace	int	No	No
PackingSlips	PackingSlips	smallint	No	No
PathLength	PathLength	smallint	No	No
PrintQue	PrintQue	varchar(100)	No	No
PullExpirationTime	PullExpirationTime	float	No	No
PullFtpFileName	PullFtpFileName	varchar(255)	No	No
PullFtpHost	PullFtpHost	varchar(20)	No	No
PullFtpPassword	PullFtpPassword	char(8)	No	No
PullFtpUserName	PullFtpUserName	char(8)	No	No
RequestsProgressPath	RequestsProgressPath	varchar(255)	No	No
ResourceQueuePath	ResourceQueuePath	varchar(255)	No	No
Retries	Retries	smallint	No	No
Rootpath	Rootpath	varchar(255)	No	No
ServerHandle	ServerHandle	varchar(255)	No	No
ServerId	ServerId	varchar(50)	Yes	Yes
ServerType	ServerType	varchar(20)	No	Yes

Name	Code	Type	PK	Mandatory
Sleepetime	Sleepetime	smallint	No	No
Status	Status	char(9)	No	No
TempDirectory	TempDirectory	varchar(255)	No	No
TimeFactor	TimeFactor	float	No	No
TimerDuration	TimerDuration	smallint	No	No
TimerId	TimerId	int	No	No
TimerReload	TimerReload	smallint	No	No
TotalSpace	TotalSpace	int	No	No
UserStagingSpace	UserStagingSpace	int	No	No

**Table: DsStDeleteLogFile**

### Description

Contains a historic record of each file that Storage Management deletes from its (Pull Monitor) cache (or the DsStCacheFile table). This entity will be used for maintaining a history of file usage and cache usage for reporting and analysis purposes. An entry is inserted into the entity via a delete trigger on the DsStCacheFile table.

### Column List

Name	Code	Type	PK	Mandatory
ActiveLinkCount	ActiveLinkCount	smallint	No	No
AlwaysInCache	AlwaysInCache	boolean	No	No
CacheHitCount	CacheHitCount	smallint	No	No
DeleteDate	DeleteDate	datetime	No	Yes
DeleteLogId	DeleteLogId	int	Yes	Yes
ExpirationFlag	ExpirationFlag	char(11)	No	No
FileName	FileName	varchar(100)	No	No
FileSize	FileSize	int	No	No
FileWeight	FileWeight	float	No	No
OriginalFileName	OriginalFileName	varchar(100)	No	No
TimeOfCachePlacement	TimeOfCachePlacement	datetime	No	No
TimeOfLastUse	TimeOfLastUse	datetime	No	No

**Table: DsStDevice**

### Description

Contains an entry for each peripheral device that Storage Management uses to service requests to Ingest or Distribute data.

## Column List

Name	Code	Type	PK	Mandatory
Capacity	Capacity	int	No	No
CurrentOperation	CurrentOperation	smallint	No	No
CurrentStatus	CurrentStatus	smallint	No	No
Description	Description	char(100)	No	No
DeviceName	DeviceName	varchar(20)	Yes	Yes
DriveCurrentSlot	DriveCurrentSlot	char(10)	No	No
DriveNumber	DriveNumber	smallint	No	No
DriveTapeld	DriveTapeld	int	No	No
ElementNo	ElementNo	smallint	No	No
FilePartitionPath	FilePartitionPath	varchar(255)	No	No
Model	Model	char(20)	No	No
Node	Node	varchar(20)	No	No
OperationStatus	OperationStatus	smallint	No	No
PathName	PathName	varchar(255)	No	No
ServerId	ServerId	varchar(50)	No	Yes
StackerId	StackerId	varchar(20)	No	No
Status	Status	smallint	No	Yes

## Table: DsStErrorAttribute

### Description

Required for all clients which wish to use the DsStErrorDetails class. Provides a mapping between character mnemonics and numeric error codes. It also defines the attributes for each error, providing adequate characterization for clients to infer appropriate retry/recovery procedures from the error attributes.

## Column List

Name	Code	Type	PK	Mandatory
ErrorCode	ErrorCode	smallint	Yes	Yes
Mnemonic	Mnemonic	char(32)	No	Yes
Scope	Scope	char(1)	No	No
Severity	Severity	char(1)	No	No

## Table: DsStErrorText

### Description

Provides text descriptions and suggested recovery actions for each error code; presents errors in a meaningful manner.

## Column List

Name	Code	Type	PK	Mandatory
Description	Description	char(255)	No	No
Mnemonic	Mnemonic	char(32)	Yes	Yes
Suggestion	Suggestion	varchar(255)	No	No

## Table: DsStEventLog

### Description

Contains a history of events and COTS errors encountered by Storage Management. The STMGT software will insert records into the table using the stored procedure DsStELInsert.sp. The calling sequence is: DsStELInsert @EventNumber=value, @EventMessage=value, @EventDate=value, @EventType=value.

The Storage Management software will insert a new ERROR\_LOG entry each time an event occurs or an error is encountered. The operator will have the ability to purge this entity periodically based on a date/time value.

## Column List

Name	Code	Type	PK	Mandatory
EventDate	EventDate	datetime	No	Yes
EventLevel	EventLevel	varchar(11)	No	No
EventLogId	EventLogId	int	Yes	Yes
EventMessage	EventMessage	varchar(255)	No	Yes
EventNumber	EventNumber	int	No	No
EventType	EventType	varchar(10)	No	Yes

## Table: DsStFileLocation

### Description

Maintains the location(s) in a cache of each file recorded in the DsStCacheFile table. As each file is retrieved from the archive and written to one of the Storage Management caches, an entry is inserted in the entity. The entry must correspond to an existing DsStCacheFile entry.

Upon insertion of a DsStFileLocation record, the corresponding DsStCacheFile record will be updated by incrementing or decrementing the ActiveLinkCount via a Trigger and setting the LastAccessDate to the current system date and time.

## Column List

Name	Code	Type	PK	Mandatory
DirectoryId	DirectoryId	smallint	Yes	Yes
FileName	FileName	varchar(100)	Yes	Yes

## **Table: DsStNextId**

### **Description**

Used to track the automatic, sequential generation of unique identifying keys for various tables in the Storage Management database (DsStErrorLog, DsStDeleteLogCacheFile, DsStBackupHistory, DsStRestoreHistory, and DsStConfigParameter.TimerId)

### **Column List**

Name	Code	Type	Pk	Mandatory
NextId	NextId	int	No	No
TableName	TableName	varchar(30)	Yes	Yes

## **Table: DsStOffsite**

### **Description**

For mapping the offsite identifiers (ie: GSF, ERC, ...) with their appropriate Cell location (ie: gsfc.nasa.com) as well as identifying their specified Archive directory for (remote) backup to.

### **Column List**

Name	Code	Type	Pk	Mandatory
CellName	CellName	varchar(20)	No	No
CreateDate	CreateDate	datetime	Yes	Yes
EndDate	EndDate	datetime	No	No
Offsiteld	Offsiteld	char(3)	Yes	Yes
HWCI	HWCI	char(12)	No	No
ServerId	ServerId	varchar(50)	No	No
VolumeGroupName	VolumeGroupName	varchar(6)	No	No
VolumeStartDate	VolumeStartDate	datetime	No	No

## **Table: DsStSchedule**

### **Description**

Contains an entry for scheduled requests of a peripheral device that Storage Management receives.

An entry is inserted into the entity every time a peripheral device is reserved to service a request to Ingest or Distribute data. The unit of measure for the estimated duration is minutes.

## Column List

Name	Code	Type	PK	Mandatory
Comment	Comment	char(100)	No	No
DeviceName	DeviceName	varchar(20)	No	No
EndDate	EndDate	datetime	No	Yes
EstimatedDuration	EstimatedDuration	smallint	No	No
EstimatedEndDate	EstimatedEndDate	datetime	No	Yes
EstimatedStartDate	EstimatedStartDate	datetime	No	Yes
Operator	Operator	varchar(20)	No	No
Priority	Priority	smallint	No	No
RecdDate	RecdDate	datetime	No	No
RequestId	RequestId	char(20)	No	No
ScheduleId	ScheduleId	int	Yes	Yes
ServerHandle	ServerHandle	varchar(255)	No	No
ServerId	ServerId	varchar(50)	No	No
SlotNumber	SlotNumber	smallint	No	No
StackerId	StackerId	varchar(20)	No	No
StartDate	StartDate	datetime	No	Yes
Status	Status	smallint	No	No

## Table: DsStServerType

### Description

Contains all types of servers administered and configured by Storage Management and their associated description.

12 Types of standard servers are currently pre-populated with the construction of the database due to no user interface currently existing to administer (i.e.: Archive, Staging Monitor, Pull Monitor, CDR, Distribution FTP, 4MM Tape, 8MM Tape, 9 Track, Staging Disk, Ingest FTP, D3, 3480/3490, ...)

## Column List

Name	Code	Type	PK	Mandatory
PollingRequired	PollingRequired	char(1)	No	No
ServerDescription	ServerDescription	char(100)	No	No
ServerType	ServerType	varchar(20)	Yes	Yes

## Table: DsStSlot

### Description

Records each instance of a Device Stacker's available hardware slots.

## Column List

Name	Code	Type	PK	Mandatory
ElementNo	ElementNo	smallint	No	No
SlotCapacity	SlotCapacity	int	No	No
SlotCurrentDrive	SlotCurrentDrive	smallint	No	No
SlotCurrentStatus	SlotCurrentStatus	smallint	No	No
SlotNumber	SlotNumber	smallint	Yes	Yes
SlotStatus	SlotStatus	smallint	No	No
SlotTapeld	SlotTapeld	varchar(50)	No	No
SlotUse	SlotUse	smallint	No	No
StackerId	StackerId	varchar(20)	Yes	Yes

## Table: DsStStacker

### Description

Records each instance of a Stacker Device. Related to 4MM ,8MM , and D3 Tape Devices.

## Column List

Name	Code	Type	PK	Mandatory
ErrorCount	ErrorCount	smallint	No	No
FixedSlot	FixedSlot	int	No	No
Mediumtype	MediumType	varchar(20)	No	No
OnLineDrives	OnLineDrives	smallint	No	No
OnLineSlots	OnLineSlots	smallint	No	No
StackerId	StackerId	varchar(20)	Yes	Yes
StackerNumber	StackerNumber	varchar(10)	No	No
StackerPath	StackerPath	varchar(255)	No	No
TotalDrives	TotalDrives	smallint	No	No
TotalRoSlots	TotalRoSlots	smallint	No	No
TotalRwSlots	TotalRwSlots	smallint	No	No
ElementNo	ElementNo	smallint	No	No
StackerStatus	StackerStatus	smallint	No	No
TotalSlots	TotalSlots	smallint	No	No

## Table: DsStVolumeGroup

### Description

Contains 'volume group' (section of Archive you are dealing with) information from configuration files such as the path currently pointed to and a history of paths related ONLY to a particular Archive server type.

## Column List

Name	Code	Type	PK	Mandatory
ServerId	ServerId	varchar(50)	Yes	Yes
VolumeEndDate	VolumeEndDate	datetime	No	No
VolumeGroupName	VolumeGroupName	varchar(6)	Yes	Yes
VolumeGroupPath	VolumeGroupPath	varchar(255)	No	Yes
VolumeStartDate	VolumeStartDate	datetime	Yes	Yes

## Table: Slots

### Description

Originally implemented as the entity to contain an entry for each peripheral stacker device's slots that Storage Management uses to service requests to Ingest or Distribute data. Since this entity acts as a look-up entity for entries that exist in theRequest entity, the data in this entity will be initialized prior to software installation.

Replaced by new table DsStSlot.

## Column List

Name	Code	Type	PK	Mandatory
element_no	element_no	int	No	No
slot_current_drive	slot_current_drive	int	No	No
slot_number	slot_number	int	Yes	Yes
slot_status	slot_status	int	No	No
slot_tapeid	slot_tapeid	varchar(50)	No	No
slot_use	slot_use	int	No	No
stacker_id	stacker_id	varchar(50)	Yes	Yes

## Table: Stackers

### Description

Originally implemented as the entity to contain an entry for each peripheral stacker that Storage Management uses to service requests to Ingest or Distribute data. Since this entity acts as a look-up entity for entries that exist in theRequest entity, the data in this entity will be initialized prior to software installation.

Replaced by new table DsStStacker.

## Column List

Name	Code	Type	PK	Mandatory
element_no	element_no	int	No	No
error_count	error_count	int	No	No
medium_type	medium_type	varchar(50)	No	Yes
online_drives	online_drives	int	No	No
online_slots	online_slots	int	No	No
stacker_id	stacker_id	varchar(50)	Yes	Yes
stacker_path	stacker_path	varchar(50)	No	Yes
status	status	int	No	No
total_drives	total_drives	int	No	No
total_ro_slots	total_ro_slots	int	No	No
total_rw_slots	total_rw_slots	int	No	No
total_slots	total_slots	int	No	No

### 4.1.3 Columns

Brief definitions of each of the columns present in the database tables defined above are contained herein.

#### Column: ActiveLinkCount

##### Description

Number of links existing in the DsStFileLocation (pull\_link) table to a cache file. Automatically incremented and decremented via a delete and insert Trigger on the DsStFileLocation table.

#### Column: AlwaysInCache

##### Description

"Y"es or "N" if always in cache file.

#### Column: ArchiveId

##### Description

Relative to Archive Backup & Restore, it is external data received from SDSRV. It's value is in the format of "<HWCI>\_<mode>:<VolumeGroupName>".

#### Column: AvailableCacheSpace

##### Description

Remaining disk space allocated/ available (in the Pull Monitor). Determined at start time; could be different on start up (i.e.: 10000000).

## **Column: BackupHistoryId**

### **Description**

Unique identifier of historic record. Sequentially generated from DsStNextId table.

## **Column: BackupId**

### **Description**

Relative to Archive Backup & Restore, it is external data received from SDSRV. It's value is in the format of "<HWCI>\_<mode>:<VolumeGroupName>".

## **Column: BlockSize**

### **Description**

Specifies the byte size blocks for the remote tape device or disk to which we are writing. Platform specific (for all servers). The only certain known value is 2048 (bytes) which is used by default. 65K is the maximum value that can be specified (i.e.: 1024 for Sun, 4096 for SGI, ...)

## **Column: CacheHitCount**

### **Description**

Number of time cache file hit.

## **Column: CacheId**

### **Description**

Unique identifier for a cache (for PullMonitor and StagingMonitor). Currently defaulted to CacheId of 1.

## **Column: CacheReplacement**

### **Description**

Turns On or Off cache replacement.

## **Column: CacheSpace**

### **Description**

Amount of RO cache space allocated/ available on disk (to the Staging Monitor/ Disk) in blocksize increments (i.e.: 7500000).

## **Column: Capacity**

### **Description**

Total amount of space available for (Distribution and Ingest FTP, 4MM and 8MM Tapes) utilization (i.e.: 1200000000).

## **Column: CellName**

### **Description**

Related to an Offsite Id, it is the address of the remote site (i.e.: gsfc.nasa.gov)

## **Column: Comment**

### **Description**

Any record of information needing to be kept related to a scheduled activity.

## **Column: ConfirmDelete**

### **Description**

Flag whether to automatically Delete upon reaching PullExpirationTime (Pull Monitor and Staging Monitor/ Disk).

## **Column: CreateDate**

### **Description**

Part of the concatenated primary key, it is the date/time at which a current record of an Offsite Id is created. Used for historical reference to modifications made to an Offsite Id record.

## **Column: CurrentOperation**

### **Description**

Operation of the device at the present which can be one of Read, Write, Null

## **Column: CurrentStatus**

### **Description**

Can be of Inprogress, Pending, Complete

## **Column: Datalist**

### **Description**

Path and name of (Staging Monitor/ Disk) list of files in cache area (i.e.: /home/dsst/pull.list).

**Column: DefaultBlockFactor****Description****Column: DeleteDate****Description**

Date file removed from cache.

**Column: DeleteLogId****Description**

Unique identifier of record in the log.

**Column: Description****Description**

Description of any peripheral device or entity related to Storage Management functionality (i.e.: Large capacity 4MM Tape stacker device; Unique Pull Monitor Cache)

**Column: DeviceName****Description**

Unique identifying name of device (i.e. 4MM2 Tape).

**Column: DeviceTablePath****Description****Column: Directory****Description**

Created (by Pull Monitor) when files are linked and FTP'd (i.e.: /home/dsst/user).

**Column: DirectoryId****Description**

Unique identifier of cache directory.

**Column: DirectoryPath****Description**

Full directory path in which cache is located.

## **Column: drive\_current\_slot**

### **Description**

Current slot of tape that is in this drive.

## **Column: drive\_number**

### **Description**

Which drive of the available drives in the stacker hardware is being used (i.e.: 1, 2, or 3, ...).

## **Column: drive\_path**

### **Description**

Directory path of device/ resource files used (i.e.: /home/pakkinen).

## **Column: drive\_status**

### **Description**

Status of server (i.e. 0 for On-line or 1 for Off-line).

## **Column: drive\_tapeid**

### **Description**

Identification or file name of tape currently in this drive.

## **Column: DriveCurrentSlot**

### **Description**

Current slot of tape that is in this drive.

## **Column: DriveNumber**

### **Description**

Which drive of the available drives in the stacker hardware is being used (i.e.: 1, 2, or 3, ...).

## **Column: DriveTapeId**

### **Description**

Identification or file name of tape currently in this drive.

## **Column: EC\_KFTP\_Command**

### **Description**

If null, server code uses default from CCS library (Distribution Ftp).

## **Column: ElementNo**

### **Description**

An identifying number (for robotic arm) to find location of a device drive or slot.

## **Column: element\_no**

### **Description**

An identifying number (for robotic arm) to find location of a drive device.

## **Column: EndDate**

### **Description**

Date & time scheduled activity has ended.

## **Column: ErrorCode**

### **Description**

Unique identifier of a DsStErrorAttribute record.

## **Column: error\_count**

### **Description**

Count of number of errors received on stacker.

## **Column: ErrorCount**

### **Description**

Count of number of errors received on stacker.

## **Column: EstimatedDuration**

### **Description**

Total estimated time for duration of scheduled activity.

## **Column: EstimatedEndDate**

### **Description**

Estimated date & time at which scheduled activity will end.

## **Column: EstimatedStartDate**

### **Description**

Estimated date & time at which scheduled activity will begin.

## **Column: EventDate**

### **Description**

Date & time of occurrence of (table insertion) error/ event on event log. Supplied by calling the application.

## **Column: EventLevel**

### **Description**

(i.e.: Error, Warning, Information)

## **Column: EventLogId**

### **Description**

Unique identifier of error/ event entry on event log. Automatically generated, sequential number (insert Trigger).

## **Column: EventMessage**

### **Description**

The associated text for STMGT event or COTS error message.

## **Column: EventNumber**

### **Description**

Created independently of the Database; Number associated with STMGT Event message or COTS error number.

## **Column: EventType**

### **Description**

Categorization of error/ event on event log (i.e.: Server, Pull Monitor, Cache, Sybase, ...).

## **Column: ExpirationFlag**

### **Description**

Mark file(s) when the difference between current system datetime & DsStCacheFile.TimeOfLastUse exceeds DsStConfigParameter.PullExpirationTime (i.e.: EXPIRED vs NOT EXPIRED).

## **Column: ExpireNotify**

### **Description**

For PullMonitorServer; 0 (true), 1 (false).

## **Column: FileName**

### **Description**

Relative to PullMonitor caching, it is the unique identifier of the metadata file (uniq\_file). Relative to Archive Backup & Restore, it is the unique identifier of the metadata file provided as external data from SDSRV.

## **Column: FilePartitionPath**

### **Description**

## **Column: FileSize**

### **Description**

The size of the cache file in bytes. File sizes of greater than 2 gigabytes are expected.

## **Column: FileWeight**

### **Description**

Weight of cache file with floating decimal.

## **Column: FixedSlot**

### **Description**

Fixed slot of the Stacker

## **Column: FullPullMonitor**

### **Description**

At what percentage will PullMonitorServer maximize capacity.

## **Column: HighCapacity**

### **Description**

Highest amount of space available for utilization ( Disk Ftp, Ingest Ftp, 8MM Tape, 4MM Tape )

## **Column: HighWaterMark**

### **Description**

Highest allowable percentage usage of space allocated (for the Pull Monitor and Staging Monitor/ Disk) (i.e. 75.80).

## **Column: LowCapacity**

### **Description**

Lowest amount of space available for utilization ( Disk Ftp , Ingest Ftp, 8MM Tape, 4MM Tape )

## **Column: LowWaterMark**

### **Description**

Delete down to limit when High Water Mark is reached (for the Pull Monitor and Staging Monitor/ Disk) (i.e. 25.50).

## **Column: MaxCommandLine**

### **Description**

Maximum length of UNIX filepath/ filename (for the Pull Monitor and Staging Monitor/ Disk) (i.e.: 300).

## **Column: MaxFiles**

### **Description**

Maximum number of files allowed in (Pull Monitor and Staging Monitor/ Disk) staging DataList file (i.e.: 250000).

## **Column: medium\_type**

### **Description**

Medium of the storage media specified for a Stacker (i.e.: 8mm Tape, 4mm Tape, D3 Tape)

## **Column: MediumType**

### **Description**

Medium of the storage media specified for a Stacker (i.e.: 8mm Tape, 4mm Tape, D3 Tape)

## **Column: Mnemonic**

### **Description**

Defined by internal standards for symbolic constants (e.g. DsCSt\*).

## **Column: Model**

### **Description**

Particular model number or reference for this device (for 4MM, 8MM Tapes and Stackers) (i.e.: CM1).

## **Column: NextId**

### **Description**

Automatically incremented and used as the next available integer for a unique identifier.

## **Column: Node**

### **Description**

The Node on which the server is currently running when it is brought up (i.e.: kodiak, soe2sun, dss2, etc.). When the server is taken down, the node field will be set to NULL (default) or blank. Automatically set by application software.

## **Column: NumColumns**

### **Description**

The number of columns for formatting a page to print. Required for the EcDsStPrintServer server.

## **Column: NumRows**

### **Description**

The number of rows for formatting a page to print. Required for the EcDsStPrintServer server configuration.

## **Column: OffsiteId**

### **Description**

Relative to Archive Backup & Restore, it is received as external data from SDSRV. It's value must be in the format of 3 characters (ie: GSF, ERC, ...).

## **Column: online\_drives**

### **Description**

Number of drives that are on-line in this stacker versus off-line.

## **Column: online\_slots**

### **Description**

Number of slots that are on-line in this stacker versus off-line.

## **Column: OnLineDrives**

### **Description**

Number of drives that are on-line in this stacker versus off-line.

## **Column: OnLineSlots**

### **Description**

Number of slots that are on-line in this stacker versus off-line.

## **Column: OperationStatus**

### **Description**

Status of Operation which can be one of Free, Allocated, Mounted, Completed

## **Column: Operator**

### **Description**

The name of the person doing the scheduling.

## **Column: OriginalCacheSpace**

### **Description**

Amount of space allocated (to the Pull Monitor) in Blocksize increments (i.e. 10000000).

## **Column: OriginalFileName**

### **Description**

Relative to PullMonitor caching, it is the original file name received from the satellite transmitted data.

Relative to Archive Backup & Restore, it is the original file name received as external data from SDSRV.

## **Column: PackingSlips**

### **Description**

The number of packing slips to be printed. Required for the EcDsStPrintServer server configuration.

## **Column: PathLength**

### **Description**

Length of UNIX filepath/ filename used in system call (for the Pull Monitor and Staging Monitor/ Disk) (i.e.: 300).

## **Column: PathName**

### **Description**

Directory path of device/ resource files used (i.e.: /home/pakkinen).

## **Column: PollingRequired**

### **Description**

'Poll' the database to see if there is any work for this server (i.e.: Y(es) or N(o)).

## **Column: PrintQue**

### **Description**

The destination printer to which to print. Required for the EcDsStPrintServer server configuration.

## **Column: Priority**

### **Description**

Priority can be of High, Medium, Low

## **Column: PullExpirationTime**

### **Description**

Duration in hours when files may be considered for deletion (PullMonitorServer) (i.e. 24).

**Column: PullFtpFileName****Description****Column: PullFtpHost****Description**

Name of host where server runs (Distribution Ftp and Archive) (i.e.: osaka).

**Column: PullFtpPassword****Description**

Password for (Distribution FTP and Archive Backup Restore) account (i.e.: user1).

**Column: PullFtpUserName****Description**

User name for (Distribution FTP and Archive Backup Restore) account (i.e.: newuser).

**Column: RecdDate****Description**

Date and time of when the activity was scheduled.

**Column: RequestId****Description**

Received from the DDIST (Data Distribution) CI.

**Column: RequestsProgressPath****Description****Column: ResourceQueuePath****Description**

Required for the Device/Resource configuration.

**Column: Retries****Description**

Number of retries if (Distribution and Ingest FTP) request fails (i.e.: 1).

## **Column: Rootpath**

### **Description**

Path to Device/ Resource (staging area cache) (for the Staging Monitor/ Disk) (i.e.: /home/dsst/pullmonitor)

## **Column: ScheduleId**

### **Description**

Unique identifier of scheduled activity.

## **Column: Scope**

### **Description**

Values equal 'R' equest, 'A' application, 'S' ystem, 'E' nterprise

## **Column: ServerDescription**

### **Description**

Description of server type (i.e.: Large capacity 4MM Tape stacker device).

## **Column: ServerHandle**

### **Description**

UNIX pathname to where server is running in DCE; CDS path. Tells how to connect to a particular server. One handle exists per Server which can relate to multiple Devices.

## **Column: ServerId**

### **Description**

Unique identifying name of server using pre-defined naming convention (i.e.: EcDsStArch--for archive server).

## **Column: ServerType**

### **Description**

Unique identifier of type of server (i.e.: Pull Monitor, Archive, Distribution FTP, D3, 4MM TAPE, ...)

## **Column: Severity**

### **Description**

Values equal 'F' alat, 'E' rror, 'W' arning

## **Column: SleepTime**

### **Description**

Duration in minutes to wait between retries (for Distribution and Ingest FTP) (i.e.: 10).

## **Column: slot\_current\_drive**

### **Description**

Current drive of tape that came from this slot.

## **Column: slot\_number**

### **Description**

Which slot of the available slots in the stacker hardware is being used (i.e.: 1, 2, or 3, ...).

## **Column: slot\_status**

### **Description**

Status of currently used slot (i.e. 0 for not in use or 1 for in use).

## **Column: slot\_tapeid**

### **Description**

Identification or file name of tape currently in this slot.

## **Column: slot\_use**

### **Description**

0 for read-only slot or 1 for read-write slot.

## **Column: SlotCapacity**

### **Description**

## **Column: SlotCurrentDrive**

### **Description**

Current drive of tape that came from this slot.

## **Column: SlotCurrentStatus**

### **Description**

0 = Off-Line or Free; 1 = On-Line or Allocated

## **Column: SlotNumber**

### **Description**

Which slot of the available slots in the stacker hardware is being used (i.e.: 1, 2, or 3, ...).

## **Column: SlotStatus**

### **Description**

Status of currently used slot (i.e. 0 for not in use or 1 for in use).

## **Column: SlotTapeId**

### **Description**

Identification or file name of tape currently in this slot.

## **Column: SlotUse**

### **Description**

0 for read-only slot or 1 for read-write slot.

## **Column: stacker\_id**

### **Description**

Unique (Medium) type and (model) number of the stacker hardware (i.e. 4MMEXB218).

## **Column: StackerId**

### **Description**

Unique (Medium) type and (model) number of the stacker hardware (i.e. 4MMEXB218).

## **Column: stacker\_path**

### **Description**

Directory path of device/ resource files used.

## **Column: StackerNumber**

### **Description**

An identifier for a Stacker such as “Stacker1\_OPS”

## **Column: StackerPath**

### **Description**

Directory path of device/ resource files used.

## **Column: StartDate**

### **Description**

Date & time at which a scheduled activity begins.

## **Column: Status**

### **Description**

Current operating status of a server (i.e. 0 for On-line or 1 for Off-line). Set to On-Line by default.

## **Column: Suggestion**

### **Description**

Handling comments of recovery procedures related to an experienced error (mnemonic).

## **Column: TableName**

### **Description**

Unique name of database table using the next available id.

## **Column: TempDirectory**

### **Description**

Captured from DsStConfigParameter.ServerId, TempDirectory based on the associated ArchiveId of the Backup file.

## **Column: TimeFactor**

### **Description**

Multiplied by (file) bytes; gives an estimated transfer rate

## **Column: TimeOfCachePlacement**

### **Description**

Date and time cache file is created.

## **Column: TimeOfLastUse**

### **Description**

Date and time cache file last used.

## **Column: TimerDuration**

### **Description**

Time interval to delete the files (FtpNotifyFileName) that are Ftp'd.

## **Column: TimerId**

### **Description**

If DsStServerType.PollingRequired set to Y(es) then TimerId automatically generated otherwise null.

## **Column: TimerReload**

### **Description**

## **Column: total\_drives**

### **Description**

Total number of drives/ devices available in this stacker (for Distribution and Ingest FTP, 4MM and 8MM Tapes) (i.e. 1, 2, ...).

## **Column: total\_ro\_slots**

### **Description**

Number of (cartridge) slots that are Read-only.

## **Column: total\_rw\_slots**

### **Description**

Number of (cartridge) slots that are Read-write.

## **Column: total\_slots**

### **Description**

Total number of slots available in this stacker (i.e.: 10, 12, 20, ...)

## **Column: TotalDrives**

### **Description**

Total number of drives/ devices available in this stacker (for Distribution and Ingest FTP, 4MM and 8MM Tapes) (i.e. 1, 2, ...).

## **Column: TotalRoSlots**

### **Description**

Number of (cartridge) slots that are Read-only.

## **Column: TotalRwSlots**

### **Description**

Number of (cartridge) slots that are Read-write.

## **Column: TotalSlots**

### **Description**

Total number of slots available in this stacker (i.e.: 10, 12, 20, ...)

## **Column: TotalSpace**

### **Description**

The total size of raid allocated (to Staging Monitor and Staging Disk combined) (i.e.: 10000000).

## **Column: TransferStage**

### **Description**

Values, in logical order, include first, 'secondary', second, 'temporary', third, 'offsite'.

## **Column: TransferStatus**

### **Description**

Values, in logical order, include first, "executing" and second either "succeeded" or "failed".

## **Column: UserStagingSpace**

### **Description**

Amount of space dedicated (to the Staging Disk). Value calculated as the difference between Total\_Space and Cache\_Space values (i.e.: 2500000).

## **Column: VolumeEndDate**

### **Description**

Date & time volume group use completed.

## **Column: VolumeGroupName**

### **Description**

Name of volume group (i.e.: VG1).

## **Column: VolumeGroupPath**

### **Description**

Location of files associated with volume group (i.e.: /amass/volumegroupone).

## **Column: VolumeStartDate**

### **Description**

Date & time volume group use created.

### **4.1.4 Column Domains**

Domains specify the ranges of values allowed for a given table column. Sybase supports the definition of specific domains to further limit the format of data for a given column. Sybase domains are, in effect, user-defined data types. The domains defined in the STMGMT database are given here as well as an indication of the columns to which they apply.

## **Domain: backup**

### **Description**

## Reference List

Table Code	Column Name	Column Code
DsStBackupHistory	Archiveld	Archiveld
DsStBackup	Archiveld	Archiveld
DsStBackupHistory	BackupId	BackupId
DsStBackup	BackupId	BackupId

## Domain: blocksize

### Description

## Reference List

Table Code	Column Name	Column Code
DsStConfigParameter	BlockSize	BlockSize

## Domain: boolean

### Reference List

Table Code	Column Name	Column Code
DsStDeleteLogCacheFile	AlwaysInCache	AlwaysInCache
DsStConfigParameter	ConfirmDelete	ConfirmDelete
DsStConfigParameter	ExpireNotify	ExpireNotify

## Domain: cacheid

### Reference List

Table Code	Column Name	Column Code
DsStCacheDirectory	Cacheld	Cacheld
DsStConfigParameter	Cacheld	Cacheld
DsStCache	Cacheld	Cacheld

## **Domain: capacity**

### **Reference List**

Table Code	Column Name	Column Code
DsStDevice	Capacity	Capacity
DsStConfigParameter	HighCapacity	HighCapacity
DsStConfigParameter	LowCapacity	LowCapacity
DsStSlot	SlotCapacity	SlotCapacity

## **Domain: cell**

## **Domain: datalist**

### **Reference List**

Table Code	Column Name	Column Code
DsStConfigParameter	Datalist	Datalist

## **Domain: defaultblockfactor**

### **Reference List**

Table Code	Column Name	Column Code
DsStConfigParameter	DefaultBlockFactor	DefaultBlockFactor

## **Domain: description**

### **Reference List**

Table Code	Column Name	Column Code
DsStSchedule	Comment	Comment
DsStDevice	Description	Description
DsStCache	Description	Description
DsStServerType	ServerDescription	ServerDescription

## **Domain: devicename**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStDevice	DeviceName	DeviceName
DsStSchedule	DeviceName	DeviceName
DsStStacker	StackerId	StackerId
DsStSlot	StackerId	StackerId

## **Domain: directory**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStConfigParameter	Directory	Directory

## **Domain: directoryid**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStFileLocation	DirectoryId	DirectoryId
DsStCacheDirectory	DirectoryId	DirectoryId

## **Domain: drivenumber**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStDevice	DriveNumber	DriveNumber

## **Domain: drivetapeid**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStDevice	DriveTapeld	DriveTapeld

## **Domain: duration**

### **Reference List**

Table Code	Column Name	Column Code
DsStSchedule	EstimatedDuration	EstimatedDuration
DsStConfigParameter	TimerDuration	TimerDuration
DsStConfigParameter	TimerReload	TimerReload

## **Domain: elementno**

### **Reference List**

Table Code	Column Name	Column Code
DsStDevice	ElementNo	ElementNo
DsStStacker	ElementNo	ElementNo
DsStSlot	ElementNo	ElementNo

## **Domain: eventlevel**

### **Reference List**

Table Code	Column Name	Column Code
DsStEventLog	EventLevel	EventLevel

## **Domain: eventmessage**

### **Reference List**

Table Code	Column Name	Column Code
DsStEventLog	EventMessage	EventMessage

## **Domain: eventnumber**

### **Reference List**

Table Code	Column Name	Column Code
DsStEventLog	EventNumber	EventNumber

## **Domain: eventtype**

### **Reference List**

Table Code	Column Name	Column Code
DsStEventLog	EventType	EventType

## **Domain: expirationflag**

### **Reference List**

Table Code	Column Name	Column Code
DsStCacheFile	ExpirationFlag	ExpirationFlag
DsStDeleteLogCacheFile	ExpirationFlag	ExpirationFlag

## **Domain: file**

### **Reference List**

Table Code	Column Name	Column Code
DsStCacheFile	FileName	FileName
DsStFileLocation	FileName	FileName
DsStDeleteLogCacheFile	FileName	FileName
DsStBackupHistory	FileName	FileName
DsStCacheFile	OriginalFileName	OriginalFileName
DsStBackupHistory	OriginalFileName	OriginalFileName
DsStDeleteLogCacheFile	OriginalFileName	OriginalFileName
DsStConfigParameter	PrintQue	PrintQue

## **Domain: filesize**

### **Reference List**

Table Code	Column Name	Column Code
DsStCacheFile	FileSize	FileSize
DsStDeleteLogCacheFile	FileSize	FileSize

## **Domain: fileweight**

### **Reference List**

Table Code	Column Name	Column Code
DsStCacheFile	FileWeight	FileWeight
DsStDeleteLogCacheFile	FileWeight	FileWeight

## **Domain: flag**

### **Reference List**

Table Code	Column Name	Column Code
DsStCacheFile	AlwaysInCache	AlwaysInCache
DsStConfigParameter	CacheReplacement	CacheReplacement
DsStServerType	PollingRequired	PollingRequired

## **Domain: full**

### **Reference List**

Table Code	Column Name	Column Code
DsStConfigParameter	FullPullMonitor	FullPullMonitor

## **Domain: id**

### **Reference List**

Table Code	Column Name	Column Code
DsStDeleteLogCacheFile	DeleteLogId	DeleteLogId
DsStEventLog	EventLogId	EventLogId
DsStNextId	NextId	NextId
DsStSchedule	ScheduleId	ScheduleId

## **Domain: maxcommandline**

### **Reference List**

Table Code	Column Name	Column Code
DsStConfigParameter	MaxCommandLine	MaxCommandLine

## **Domain: maxfiles**

### **Reference List**

Table Code	Column Name	Column Code
DsStConfigParameter	MaxFiles	MaxFiles

## **Domain: model**

### **Reference List**

Table Code	Column Name	Column Code
DsStDevice	Model	Model

## **Domain: node**

### **Reference List**

Table Code	Column Name	Column Code
DsStDevice	Node	Node
DsStConfigParameter	Node	Node
DsStConfigParameter	PullFtpHost	PullFtpHost

## **Domain: offsite**

### **Reference List**

Table Code	Column Name	Column Code
DsStOffsite	Offsiteld	Offsiteld

## **Domain: operator**

### **Reference List**

Table Code	Column Name	Column Code
DsStSchedule	Operator	Operator

## **Domain: password**

### **Reference List**

Table Code	Column Name	Column Code
DsStConfigParameter	PullFtpPassword	PullFtpPassword

## **Domain: path**

### **Reference List**

Table Code	Column Name	Column Code
DsStCacheDirectory	DirectoryPath	DirectoryPath
DsStConfigParameter	EC_KFTP_Command	EC_KFTP_Command
DsStDevice	FilePartitionPath	FilePartitionPath
DsStDevice	PathName	PathName
DsStConfigParameter	Rootpath	Rootpath
DsStConfigParameter	ServerHandle	ServerHandle
DsStSchedule	ServerHandle	ServerHandle
DsStStacker	StackerPath	StackerPath
DsStConfigParameter	TempDirectory	TempDirectory
DsStVolumeGroup	VolumeGroupPath	VolumeGroupPath

## **Domain: pathlength**

### **Reference List**

Table Code	Column Name	Column Code
DsStConfigParameter	PathLength	PathLength

## **Domain: priority**

### **Reference List**

Table Code	Column Name	Column Code
DsStSchedule	Priority	Priority

## **Domain: requestid**

### **Reference List**

Table Code	Column Name	Column Code
DsStSchedule	RequestId	RequestId

## **Domain: serverid**

### **Reference List**

Table Code	Column Name	Column Code
DsStDevice	ServerId	ServerId
DsStSchedule	ServerId	ServerId
DsStConfigParameter	ServerId	ServerId
DsStVolumeGroup	ServerId	ServerId
DsStOffsite	ServerId	ServerId

## **Domain: servertype**

### **Reference List**

Table Code	Column Name	Column Code
DsStConfigParameter	ServerType	ServerType
DsStServerType	ServerType	ServerType

## **Domain: sleeptime**

### **Reference List**

Table Code	Column Name	Column Code
DsStConfigParameter	Sleeptime	Sleeptime

## **Domain: slotnumber**

### **Reference List**

Table Code	Column Name	Column Code
DsStSchedule	SlotNumber	SlotNumber
DsStDevice	DriveCurrentSlot	DriveCurrentSlot
DsStSlot	SlotNumber	SlotNumber

## **Domain: slottapeid**

### **Reference List**

Table Code	Column Name	Column Code
DsStSlot	SlotTapeld	SlotTapeld

## **Domain: smallcount**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStDeleteLogFile	ActiveLinkCount	ActiveLinkCount
DsStCacheFile	ActiveLinkCount	ActiveLinkCount
DsStDeleteLogFile	CacheHitCount	CacheHitCount
DsStCacheFile	CacheHitCount	CacheHitCount
DsStStacker	ErrorCount	ErrorCount
DsStStacker	OnLineDrives	OnLineDrives
DsStStacker	OnLineSlots	OnLineSlots
DsStConfigParameter	PackingSlips	PackingSlips
DsStConfigParameter	Retries	Retries
DsStSlot	SlotCurrentDrive	SlotCurrentDrive
DsStStacker	TotalDrives	TotalDrives
DsStStacker	TotalRoSlots	TotalRoSlots
DsStStacker	TotalRwSlots	TotalRwSlots
DsStStacker	TotalSlots	TotalSlots

## **Domain: space**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStConfigParameter	AvailableCacheSpace	AvailableCacheSpace
DsStConfigParameter	CacheSpace	CacheSpace
DsStConfigParameter	OriginalCacheSpace	OriginalCacheSpace
DsStConfigParameter	TotalSpace	TotalSpace
DsStConfigParameter	UserStagingSpace	UserStagingSpace

## **Domain: state**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStConfigParameter	Status	Status

## **Domain: status**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStDevice	CurrentOperation	CurrentOperation
DsStDevice	CurrentStatus	CurrentStatus
DsStDevice	OperationStatus	OperationStatus
DsStSlot	SlotCurrentStatus	SlotCurrentStatus
DsStSlot	SlotStatus	SlotStatus
DsStSlot	SlotUse	SlotUse
DsStStacker	StackerStatus	StackerStatus
DsStDevice	Status	Status
DsStSchedule	Status	Status

## **Domain: tablename**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStNextId	TableName	TableName

## **Domain: timefactor**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStConfigParameter	PullExpirationTime	PullExpirationTime
DsStConfigParameter	TimeFactor	TimeFactor

## **Domain: timerid**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStConfigParameter	TimerId	TimerId

## **Domain: username**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStConfigParameter	PullFtpUserName	PullFtpUserName

## **Domain: volumegroupname**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStVolumeGroup	VolumeGroupName	VolumeGroupName
DsStOffsite	VolumeGroupName	VolumeGroupName

## **Domain: watermark**

### **Reference List**

<b>Table Code</b>	<b>Column Name</b>	<b>Column Code</b>
DsStConfigParameter	HighWaterMark	HighWaterMark
DsStConfigParameter	LowWaterMark	LowWaterMark

### **4.1.5 Rules**

Sybase supports the definitions of rules. Rules provide a means for enforcing domain constraints on a given column. All rules defined in Sybase for the STMGMT database are described herein.

### **4.1.6 Defaults**

Defaults are used to supply a value for a column when one is not defined at insert time. There are no defaults defined in Sybase for the STMGMT database.

### **4.1.7 Views**

Sybase allows the definition of views as a means of limiting an application or users access to data in a table or tables. Views create a logical table from columns found in one or more tables. There are no views defined in Sybase for the STMGMT database.

### **4.1.8 Integrity Constraints**

Sybase version 11.0.1 allows the enforcement of referential integrity via the use of declarative integrity constraints. Integrity constraints allow the SQL server to enforce primary and foreign key integrity checks without automatically requiring programming. Sybase 11 is only ANSI-92 compliant, however, therefore its constraints support “restrict-only” operations. This means that a row can not be deleted or updated if there are rows in other tables having a foreign key dependency on that row. Cascade delete and update operations can not be performed if a declarative constraint has been used. All declarative integrity constraints defined in the STMGMT database are described herein.

## **Dependencies on Table: DsStCache**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DsStCacheDirectory	Cacheld	Cacheld
DsStConfigParameter	Cacheld	Cacheld

## **Dependencies on Table: DsStCacheDirectory**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DsStFileLocation	DirectoryId	DirectoryId

## **Dependencies on Table: DsStCacheFile**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DsStFileLocation	FileName	FileName

## **Dependencies on Table: DsStConfigParameter**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DsStSchedule	ServerId	ServerId
DsStVolumeGroup	ServerId	ServerId
DsStDevice	ServerId	ServerId

## **Dependencies on Table: DsStDevice**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DsStSchedule	DeviceName	DeviceName

## **Dependencies on Table: DsStErrorText**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DsStErrorAttribute	Mnemonic	Mnemonic

## **Dependencies on Table: DsStOffsite**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DsStBackup	Offsiteld	Offsiteld

## **Dependencies on Table: DsStServerType**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DsStConfigParameter	ServerType	ServerType

## **Dependencies on Table: DsStSlot**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DsStSchedule	SlotNumber	SlotNumber

## **Dependencies on Table: DsStStacker**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DsStSchedule	StackerId	StackerId
DsStSlot	StackerId	StackerId
DsStDevice	StackerId	StackerId

## Dependencies on Table: DsStVolumeGroup

### Reference by List

Referenced by	Primary Key	Foreign Key
DsStConfigParameter	ServerId	ServerId
DsStOffsite	ServerId VolumeGroupName VolumeStartDate	ServerId VolumeGroupName VolumeStartDate

### 4.1.9 Triggers

Sybase supports the enforcement of business policy via the use of triggers. A trigger is best defined as set of activities or checks that should be performed automatically when ever a row is inserted, updated, or deleted from a given table. Sybase version 11.0.1 allows the definition of insert, update, and delete trigger per table. A listing of each the triggers in the STMGMT database is given here. A brief definition of each of these triggers follows.

Scripts found in directory /ecs/formal/DSS/stmgt/src/Database. Descriptions of found within leading comments of each script file.

### Trigger List

Table	Trigger	Description
DsStCacheFile	DsStCFDeleteTrig	
DsStConfigParameter	DsStCPIinsertTrigTimerId	
DsStEventLog	DsStELInsertTrig	
DsStSchedule	DsStSInsertTrig	
DsStSchedule	DsStSUpdateTrig	

### Trigger: DsStCFDeleteTrig

#### Trigger Code

```
--*****
-- BEGIN PROLOG
--
-- TRIGGER NAME:DsStCFDeleteTrig
--
-- DESCRIPTION: Delete trigger on the DsStCacheFile table
--              that logs deletions from another table
--              using a cursor if DELETE has more than 1 row.
-- DATE CREATED:10/12/97
-- CREATE BY:      Lisa Colombo
```

```

-- TABLES ACCESSED:  DsStCacheFile
--                      DsStDeleteLogCacheFile
--
-- RETURNS:          Status (Success = 0)
--
--*****CREATE TRIGGER DsStCFDeleteTrig ON DsStCacheFile
FOR DELETE AS

BEGIN
DECLARE      @mystatus      int,
            @del_rows      int,
            @NextId        id

-- If no rows deleted, exit trigger.
IF @@rowcount = 0
    RETURN

-- Initialize @Status variable to 0 to represent Success. Any other
-- value will represent a failure/error or warning status.
SELECT @mystatus = 0

-- For deleted DsStCacheFile records, insert into DsStDeleteLogCacheFile.
SELECT @del_rows = count(*) from deleted

-- Trigger fired by delete.
BEGIN
    EXEC @mystatus = DsStNextIdGet "DsStDeleteLogCacheFile", @NextId OUTPUT
    INSERT DsStDeleteLogCacheFile
        SELECT @NextId,
               FileName,
               CacheId,
               FileSize,
               FileWeight,
               TimeOfCachePlacement,
               TimeOfLastUse,
               CacheHitCount,
               ActiveLinkCount,
               AlwaysInCache,
               getdate(),
               OriginalFileName,
               ExpirationFlag
        FROM deleted
    END
    RETURN

IF (@@error != 0) or (@mystatus != 0)
BEGIN
    ROLLBACK TRANSACTION
    SELECT @mystatus = 1
    RAISERROR 34003 "DsStCFDeleteTrig: DeleteLogId was not returned from

```

DsStNextIdGet stored procedure. DeleteLogId in the DsStDeleteLogFile table not updated."

```
    RETURN  
END  
END  
go
```

## Trigger: DsStCPInsertTrigTimerId

### Trigger Code

```
--*****  
--  
-- TRIGGER NAME:DsStCPInsertTrigTimerId  
--  
-- DESCRIPTION: Insert trigger for the DsStConfigParameter table.  
-- This trigger will generate the TimerId  
-- value for the record being inserted.  
--  
-- DATE MODIFIED: 10/28/97; LJC - modify table name  
--  
-- TABLES ACCESSED: DsStConfigParameter  
--  
-- RETURNS: Status (Success = 0)  
--*****
```

```
CREATE TRIGGER DsStCPInsertTrigTimerId  
ON DsStConfigParameter  
FOR INSERT AS  
BEGIN  
    DECLARE @NextId int,  
            @Status status  
  
    -- Initialize @Status variable to 0 to represent Success. Any other  
    -- value will represent a failure/error or warning status.  
  
    SELECT @Status = 0  
  
    -- Check to see if the Server has Polling enabled.  
    IF (SELECT PollingRequired FROM DsStServerType, inserted  
        WHERE DsStServerType.ServerType =  
              inserted.ServerType) = '0'  
    BEGIN  
        -- Server does not poll therefore, TimerId is not a required field.  
        RETURN  
    END
```

```

END

-- Get the next available id for this table.

EXEC @Status = DsStNextIdGet "DsStConfigParameter", @NextId OUTPUT

IF (@@error != 0) or (@Status != 0)
BEGIN
    ROLLBACK TRANSACTION
    SELECT @Status = 1
    RAISERROR 32050 "DsStCPIInsertTrigTimerId: TimerId was not returned from
DsStNextIdGet stored procedure. TimerId in the DsStConfigParameter table not updated."
    RETURN
END

-- If the next available id for this table was successfully returned,
-- update the current record with the next id value.

IF (@Status = 0)
BEGIN

    UPDATE DsStConfigParameter
        SET TimerId = @NextId
        WHERE TimerId = 0

    IF (@@error != 0)
    BEGIN
        ROLLBACK TRANSACTION
        RAISERROR 32051 "DsStCPIInsertTrigTimerId: Unable to update TimerId in
DsStConfigParameter."
        RETURN
    END
END

END
go

```

## **Trigger: DsStELInsertTrig**

### **Trigger Code**

```

--***** BEGIN PROLOG *****
-- BEGIN PROLOG
--
-- TRIGGER NAME:DsStELInsertTrig
--
-- DESCRIPTION: Insert trigger for the DsStEventLog table.
-- This trigger will generate the EventLogId
-- value for the record being inserted.

```

```

-- TABLES ACCESSED: DsStEventLog
-- RETURNS: Status (Success = 0)
-- ****
CREATE TRIGGER DsStELInsertTrig
ON DsStEventLog
FOR INSERT AS
BEGIN
DECLARE @NextId id,
        @Status status
-- Initialize @Status variable to 0 to represent Success. Any other
-- value will represent a failure/error or warning status.
SELECT @Status = 0
-- Get the next available id for this table.
EXEC @Status = DsStNextIdGet "DsStEventLog", @NextId OUTPUT
IF (@@error != 0) or (@Status != 0)
BEGIN
    ROLLBACK TRANSACTION
    SELECT @Status = 1
    RAISERROR 33020 "DsStELInsertTrig: EventLogId was not returned from DsStNextIdGet
stored procedure. EventLogId in the DsStEventLog table not updated."
    RETURN
END
-- If the next available id for this table was successfully returned,
-- update the current record with the next id value.
IF (@Status = 0)
BEGIN
    UPDATE DsStEventLog
        SET EventLogId = @NextId
        WHERE EventLogId = 0
    IF (@@error != 0)
    BEGIN
        ROLLBACK TRANSACTION
        RAISERROR 33019 "DsStELInsertTrig: Unable to update EventLogId in DsStEventLog."
        RETURN
    END

```

```
END  
END  
go
```

## Trigger: DsStSInsertTrig

### Trigger Code

```
--*****  
-- BEGIN PROLOG  
--  
-- TRIGGER NAME:DsStSInsertTrig  
--  
-- DESCRIPTION: Insert trigger for the DsStSchedule table.  
-- This trigger will generate the DsStScheduleId  
-- value for the record being inserted.  
--  
-- TABLES ACCESSED: DsStSchedule  
--  
-- RETURNS: Status (Success = 0)  
--*****
```

```
CREATE TRIGGER DsStSInsertTrig
```

```
ON DsStSchedule
```

```
FOR INSERT AS
```

```
BEGIN
```

```
DECLARE @NextId id,  
        @Status status,  
        @DeviceName devicename
```

```
-- Initialize @Status variable to 0 to represent Success. Any other  
-- value will represent a failure/error or warning status.
```

```
SELECT @Status = 0
```

```
SELECT @DeviceName = NULL
```

```
-- The DeviceName column is a foreign key into the DsStDevice table.  
-- Since DeviceName is not a mandatory column, a constraint cannot be  
-- utilized to enforce the referential integrity between DsStSchedule  
-- and DsStDevice (therefore, it must be enforced using a trigger).  
-- If a value for the DeviceName column is specified, verify that it  
-- is a valid device in the DsStDevice table.
```

```

SELECT @DeviceName = DeviceName
      FROM inserted

IF (@DeviceName IS NOT NULL)

BEGIN
    IF (SELECT COUNT(*)
        FROM DsStDevice, inserted
        WHERE DsStDevice.DeviceName = inserted.DeviceName) = 0
    BEGIN
        ROLLBACK TRANSACTION
        SELECT @Status = 1
        RAISERROR 31115 "DsStSInsertTrig: Device %1! not a valid DevicName in the
DsStDevice table.", @DeviceName
        RETURN
    END
END

-- Get the next available id for this table (get only if the DeviceName
-- was valid).
IF (@Status = 0)

BEGIN -- Get next id

    EXEC @Status = DsStNextIdGet "DsStSchedule", @NextId OUTPUT

    IF (@@error != 0) or (@Status != 0)
        BEGIN
            ROLLBACK TRANSACTION
            SELECT @Status = 1
            RAISERROR 31116 "DsStSInsertTrig: ScheduleId was not returned from
DsStNextIdGet stored procedure. ScheduleId in the DsStSchedule table not updated."
            RETURN
        END

    END -- Get next id

    -- If the next available id for this table was successfully returned,
    -- update the current record with the next id value.

    IF (@Status = 0)
    BEGIN

        UPDATE DsStSchedule
        SET ScheduleId = @NextId
        WHERE ScheduleId = 0

        IF (@@error != 0)
        BEGIN
            ROLLBACK TRANSACTION
            RAISERROR 31117 "DsStSInsertTrig: Unable to update ScheduleId in DsStSchedule."
        END

    END

```

```
    RETURN  
END
```

```
END
```

```
END
```

```
go
```

## Trigger: DsStSUpdateTrig

### Trigger Code

```
--*****  
-- BEGIN PROLOG  
--  
-- TRIGGER NAME:DsStSUpdateTrig  
--  
-- DESCRIPTION: Update trigger for the DsStSchedule table.  
--                 This trigger will verify that a valid device name  
--                 exists in the DsStDevice table.  
--  
-- TABLES ACCESSED: DsStSchedule  
--  
-- RETURNS: Status (Success = 0)  
--*****
```

```
CREATE TRIGGER DsStSUpdateTrig
```

```
ON DsStSchedule
```

```
FOR UPDATE AS
```

```
BEGIN
```

```
    DECLARE @Status status,  
            @DeviceName devicename
```

```
    IF UPDATE (DeviceName)
```

```
        BEGIN -- IF UPDATE
```

```
            -- Initialize @Status variable to 0 to represent Success. Any other  
            -- value will represent a failure/error or warning status.
```

```
            SELECT @Status = 0
```

```
            SELECT @DeviceName = DeviceName  
            FROM inserted
```

```

-- The DeviceName column is a foreign key into the DsStDevice table.
-- Since DeviceName is not a mandatory column, a constraint cannot be
-- utilized to enforce the referential integrity between DsStSchedule
-- and DsStDevice (therefore, it must be enforced using a trigger).
-- If a value for the DeviceName column is specified, verify that it
-- is a valid device in the DsStDevice table.

IF (SELECT COUNT(*)
    FROM DsStDevice, inserted
    WHERE DsStDevice.DeviceName = inserted.DeviceName) != 1

BEGIN
    ROLLBACK TRANSACTION
    RAISERROR 31118 "DsStSUpdateTrig: Device %1! not a valid DeviceName in the
DsStDevice table.", @DeviceName
    RETURN
END

END -- IF UPDATE

END
go

```

#### **4.1.10 Stored Procedures**

Sybase also includes support for business policy via the use of stored procedures. Stored procedures are typically used to capture a set of activities or checks that will be performed on the database repeatedly to enforce business policy and maintain data integrity. Stored procedures are parsed and complied SQL code that reside in the database and may be called by name by an application, trigger or another stored procedure. A listing of each the stored procedures in the STMGMT database is given here. A brief definition of each of these stored procedures follows.

Scripts found in directory /ecs/formal/DSS/stmgt/src/Database. Descriptions of found within leading comments of each script file.

Name	Description
DsStCFInsert	
DsStCFSelectByExpTime	
DsStCFSelectByFileName	
DsStCFSelectByNameCount	
DsStCFUpdateExpirationFlag	
DsStCInsert	
DsStCPDelete	
DsStCPIInsertArchive	
DsStCPIInsertDevice	
DsStCPIInsertDisFtp	

Name	Description
DsStCPIInsertPullMonitor	
DsStCPIInsertStagingMonitor	
DsStCPSelectByld	
DsStCPSelectByType	
DsStCPUpdate	
DsStCPUpdateArchive	
DsStCPUpdateDevice	
DsStCPUpdateDisFtp	
DsStCPUpdatePullMonitor	
DsStCPUpdateStagingMonitor	
DsStCPUpdAvailableCacheSpace	
DsStCPUpdStatusAndServerHandle	
DsStDDelete	
DsStDeAllocate	
DsStDIInsert	
DsStDSelect	
DsStDSelectAll	
DsStDSelectDeviceInfo	
DsStDSelectResPoolInfo	
DsStDUpdate	
DsStDUpdateDevice	
DsStDUpdateStatus	
DsStELAlarms	
DsStELInsert	
DsStELPurge	
DsStELSelect	
DsStELSelectAny	
DsStELSelectByTime	
DsStELSelectByType	
DsStFLSelect	
DsStNextIdGet	
DsStNextIdInsert	
DsStPLSelectByExpTime	
DsStPLUpdateExpirationFlag	
DsStScheduler	
DsStSIInsert	
DsStSKDelete	
DsStSKInsert	
DsStSKSelect	
DsStSKSelectAll	

Name	Description
DsStSKUpdate	
DsStSLDelete	
DsStSInsert	
DsStSSelect	
DsStSLUpdate	
DsStSSelect	
DsStSSelectById	
DsStSSelectByType	
DsStSSelectDeviceName	
DsStSTDelete	
DsStSTInsert	
DsStSTSelect	
DsStSTUpdate	
DsStSUpdate	
DsStSUpdateStatus	
DsStVGInsert	
DsStVGSelectById	
DsStVGSelectHistory	
DsStVGUpdate	

## Procedure: DsStCFInsert

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStCFInsert
--
-- DESCRIPTION:          Insert a record into the DsStCacheFile table.
--                      The ExpirationFlag is set with "NOT EXP"
--
-- TABLES ACCESSED:     DsStCacheFile
--
-- RETURNS:              Status (Success = 0)
--
-- INPUTS:   PARAMETER        DOMAIN
--           -----
--           FileName       file,
--           CahceId        cacheid,
--           TimeOfLastUse  datetime,
--           FileSize        filesize,
--           ActiveLinkCount smallcount,
--           OriginalFileName file
--
```

```

-- OUTPUTS:  PARAMETER          DOMAIN
-----  -----
-- ****
CREATE PROCEDURE DsStCFInsert
    @FileName      file,
    @CacheId       cacheid,
    @TimeOfLastUse datetime,
    @FileSize      filesize,
    @ActiveLinkCount smallcount,
    @OriginalFileName file
AS
BEGIN
    DECLARE @status int

    IF @FileName = NULL
    BEGIN
        RAISERROR 30014 "DsStCFInsert: FileName must be provided"
        SELECT @status = 1
    END
    ELSE
    IF @CacheId < 1
    BEGIN
        RAISERROR 30013 "DsStCFInsert: Invalid CacheId [%1!].",
                        @CacheId
        SELECT @status = 1
    END
    ELSE

    BEGIN TRANSACTION

    INSERT DsStCacheFile (
        FileName,
        CacheId,
        FileSize,
        FileWeight,
        TimeOfCachePlacement,
        TimeOfLastUse,
        CacheHitCount,
        ActiveLinkCount,
        AlwaysInCache,
        OriginalFileName,
        ExpirationFlag      /* Initialized to "NOT EXP" */
    )
    VALUES (
        @FileName,
        1,
        @FileSize,
        @TimeOfLastUse,
        @ActiveLinkCount,
        "NOT EXP"
    )

```

```

)
SELECT @status = @@error
IF (@status != 0)
BEGIN
    ROLLBACK TRANSACTION
SELECT @status = 1
RAISERROR 30018 "DsStCFInsert: Error inserting into DsStCacheFile"
    RETURN (@status)
END

COMMIT TRANSACTION
RETURN (@status)
END
go

```

## Procedure: DsStCFSelectByExpTime

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStCFSelectByExpTime
--
-- DESCRIPTION:          Retrieves all details of files for the PullMonitor
--                      that are expired.
--                      Calls DsStCFUpdateExpirationFlag stored procedure
--                      to get the details including latest expired files.
--
-- TABLES ACCESSED:     DsStCacheFile
--
-- RETURNS:              Status (Success = 0)
--                      Returns all rows in DsStCacheFile table that
--                      are in the PullExpirationTime for PullMonitor,
--                      with the ExpirationFlag as "EXPIRED"
--
-- INPUTS:   PARAMETER      DOMAIN
--           -----
--           ServeId        serverid
--
-- OUTPUTS:  PARAMETER      DOMAIN
--           -----
--           FileName       file,
--           CacheId        cacheid,
--           FileSize       filesize,
--           FileWeight     fileweight,
--           TimeOfCachePlacement  datetime,
--           TimeOfLastUse   datetime,
--           CacheHitCount   smallcount,
--           ActiveLinkCount smallcount,
--           AlwaysInCache   flag,

```

```

--      OriginalFileName      file,
--      ExpirationFlag       expirationflag
--
-- ****
CREATE PROCEDURE DsStCFSelectByExpTime
    @ServerId    serverid

AS
BEGIN

    DECLARE @status int
    DECLARE @cf_status int /* status for CacheFile Update Procedure */
    DECLARE @ExpTime int

    SELECT @status = 0
    SELECT @cf_status = 0

    IF (@ServerId = NULL OR @ServerId = "")
    BEGIN
        SELECT @status = 1
        RAISERROR 31021 "DsStCFselectByExpTime: ServerId must be provided."
    END
    ELSE

    BEGIN

        EXECUTE @cf_status = DsStCFUpdateExpirationFlag @ServerId
        IF @cf_status != 0
        BEGIN
            SELECT @status = 1
            RAISERROR 31037 "DsStCFUpdateExpirationFlag: Error in the DsStCFUpdat
eExpirationFlag. "
        END
        ELSE

        SELECT
            "FileName",          FileName,
            "CacheId",           CacheId,
            "FileSize",          FileSize,
            "FileWeight",         FileWeight,
            "TimeOfCachePlacement", TimeOfCachePlacement,
            "TimeOfLastUse",      TimeOfLastUse,
            "CacheHitCount",      CacheHitCount,
            "ActiveLinkCount",     ActiveLinkCount,
            "AlwaysInCache",       AlwaysInCache,
            "OriginalFileName",    OriginalFileName,
            "ExpirationFlag",      ExpirationFlag

        FROM  DsStCacheFile
        WHERE ExpirationFlag = "EXPIRED"
    END
END

```

```

SELECT @status = @@error
IF (@status != 0)
BEGIN
    SELECT @status = 1
    RAISERROR 34013 "DsStCFSelectByExpTime: Unable to select DsStCacheFile
records."
    RETURN(@status)
END

RETURN (@status)
END
go

```

## **Procedure: DsStCFSelectByFileName**

### **Code**

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStCFSelectByFileName
--
-- DESCRIPTION:         Select from the DsStCacheFile table for a given
--                      FileName
--
-- TABLES ACCESSED:    DsStCacheFile
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:              PARAMETER          DOMAIN
--                      -----
--                      FileName           file
--
-- OUTPUTS:             PARAMETER          DOMAIN
--                      -----
--                      FileName           file,
--                      CacheId            cacheid,
--                      FileSize           filesize,
--                      FileWeight         fileweight,
--                      TimeOfCachePlacement datetime,
--                      TimeOfLastUse       datetime,
--                      CacheHitCount      smallcount,
--                      ActiveLinkCount    smallcount,
--                      AlwaysInCache      flag,
--                      OriginalFileName   file,
--                      ExpirationFlag    expirationflag
--
-- ****

```

```
CREATE PROCEDURE DsStCFSelectByFileName
```

```

        @FileName    file
AS
BEGIN
    DECLARE @status int

    IF @FileName = NULL
    BEGIN
        RAISERROR 30014 "DsStCFSelectByFileName: FileName must be provided"
        SELECT @status = 1
    END
    ELSE
    BEGIN

        SELECT
            "FileName",          FileName,
            "CacheId",          CacheId,
            "FileSize",         FileSize,
            "FileWeight",       FileWeight,
            "TimeOfCachePlacement", TimeOfCachePlacement,
            "TimeOfLastUse",    TimeOfLastUse,
            "CacheHitCount",   CacheHitCount,
            "ActiveLinkCount", ActiveLinkCount,
            "AlwaysInCache",   AlwaysInCache,
            "OriginalFileName", OriginalFileName,
            "ExpirationFlag",  ExpirationFlag
        FROM  DsStCacheFile
        WHERE  FileName = @FileName

        SELECT @status = @@error
        IF (@status != 0)
        BEGIN
            RAISERROR 30018 "DsStCFSelectByFileName: Unable to select information about
FileName [%1!]", @FileName
            RETURN (@status)
        END

        RETURN (@status)
    END
END
go

```

## Procedure: DsStCFSelectByNameCount

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStCFSelectByNameCount
--
-- DESCRIPTION:         Select from the DsStCacheFile table for a given

```

```

--          FileName and ActiveLinkCount
--
-- TABLES ACCESSED:    DsStCacheFile
--
-- RETURNS:      Status (Success = 0)
--
-- INPUTS:      PARAMETER          DOMAIN
--              -----          -----
--              FileName        file,
--              ActiveLinkCount smallcount
--
-- OUTPUTS:     PARAMETER          DOMAIN
--              -----          -----
--              FileName        file,
--              CacheId         cacheid,
--              FileSize        filesize,
--              FileWeight      fileweight,
--              TimeOfCachePlacement datetime,
--              TimeOfLastUse   datetime,
--              CacheHitCount   smallcount,
--              ActiveLinkCount smallcount,
--              AlwaysInCache   flag,
--              OriginalFileName file,
--              ExpirationFlag expirationflag
--
-- ****
CREATE PROCEDURE DsStCFSelectByNameCount
    @FileName      file ,
    @ActiveLinkCount smallcount
AS
BEGIN
    DECLARE @status int

    IF @FileName = NULL
    BEGIN
        RAISERROR 30014 "DsStCFSelectByNameCount: FileName must be provided"
        SELECT @status = 1
    END
    ELSE
    BEGIN

        SELECT
            "FileName",          FileName,
            "CacheId",           CacheId,
            "FileSize",          FileSize,
            "FileWeight",         FileWeight,
            "TimeOfCachePlacement", TimeOfCachePlacement,
            "TimeOfLastUse",      TimeOfLastUse,
            "CacheHitCount",      CacheHitCount,
            "ActiveLinkCount",    ActiveLinkCount,
            "AlwaysInCache",      AlwaysInCache,
    
```

```

    "OriginalFileName",      OriginalFileName,
    "ExpirationFlag",       ExpirationFlag

FROM   DsStCacheFile
WHERE  FileName = @FileName
AND    ActiveLinkCount >= @ActiveLinkCount

SELECT @status = @@error
IF (@status != 0)
BEGIN
RAISERROR 30018 "DsStCFSelectByNameCount: Unable to select information about
FileName [% 1!]", @FileName
RETURN (@status)
END

RETURN (@status)
END
RETURN (@status)
END
go

```

## Procedure: DsStCFUpdateExpirationFlag

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStCFUpdateExpirationFlag
--
-- DESCRIPTION: To update the ExpirationFlag in DsStCacheFile table
--               for the PullMonitor .
--               Called from DsStCFSelectByExpTime
--
-- TABLES ACCESSED:  DsStCacheFile
--                  DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--                   Updates ExpirationFlag in DsStCacheFile table
--                   for all the rows that
--                   are in the ExpirationThreshold for PullMonitor,
--                   and sets the ExpirationFlag to "EXPIRED"
--
-- INPUTS:           PARAMETER      DOMAIN
--                   -----
--                   ServerId        serverid
--
-- OUTPUTS:          PARAMETER      DOMAIN
--                   -----
-- ****

```

```

CREATE PROCEDURE DsStCFUpdateExpirationFlag
    @ServerId      serverid
AS
BEGIN

    DECLARE @status int
    DECLARE @rows int
    DECLARE @ExpTime int

    SELECT @status = 0
    SELECT @rows = 0

    SELECT @ExpTime = PullExpirationTime from DsStConfigParameter
    WHERE ServerId = @ServerId
    IF (@@rowcount = 0)
        BEGIN
            SELECT @status = 1
            RAISERROR 31121 "DsStCFUpdateExpirationFlag: No Records for Pull Monitor
Server [%1!].", @ServerId
        END
    ELSE
        BEGIN -- BEGIN ELSE

            BEGIN TRANSACTION
            UPDATE DsStCacheFile
            SET   ExpirationFlag = "EXPIRED"
            WHERE datediff(hour, TimeOfLastUse ,getdate()) > @ExpTime

            IF (@@error != 0)
                BEGIN
                    SELECT @status = 1
                    ROLLBACK TRANSACTION
                    RAISERROR 31122 "DsStCFUpdateExpirationFlag: Error in updating. "
                END

            COMMIT TRANSACTION
        END -- END ELSE
        RETURN (@status)
    END
go

```

## Procedure: DsStCInsert

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStCInsert
--
-- DESCRIPTION:      Inserts a row into the DsStCache table.
-- CREATE DATE:      Oct. 22, 1997
-- DEVELOPER:        Lisa Colombo
--
-- TABLES ACCESSED:  DsStCache
--
-- RETURNS:          Status (Success = 0)
--
-- INPUT PARAMETERS           DOMAIN
-- -----
-- CacheId                  cacheid
-- Description               description
--
-- OUTPUT PARAMETERS          DOMAIN
-- -----
-- NONE
-- ****
CREATE PROCEDURE DsStCInsert
    @CacheId      cacheid,
    @Description   description
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0
    -- Error checking to see if CacheId is null
    IF @CacheId is null
        BEGIN
            SELECT @mystatus = 1
            raiserror 34000 "DsStCInsert: CacheId may not be null."
            RETURN (@mystatus)
        END
    -- Error checking to see if CacheId already exists
    IF EXISTS (Select 1 from DsStCache where CacheId = @CacheId)
        BEGIN
            SELECT @mystatus = 1
            raiserror 34001 "DsStCInsert: Unable to insert CacheId [%1!] as Cache already exists.", @CacheId
            RETURN @mystatus
        END
```

```

BEGIN TRANSACTION
INSERT DsStCache
    (CacheId,
     Description)
VALUES (@CacheId,
        @Description)

SELECT @mystatus = @@error

IF (@mystatus != 0)
BEGIN
    SELECT @mystatus = 1
    ROLLBACK TRANSACTION
    raiserror 34002 "DsStCInsert: Unable to insert CacheId [%1!].", @CacheId
END

COMMIT TRANSACTION
RETURN (@mystatus)
END
go

```

## Procedure: DsStCPDelete

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:   DsStCPDelete
--
-- DESCRIPTION:      Deletes a row from the DsStConfigParameter Table
-- MODIFICATION: 10/3/97; LJC - accounted for ServerId foreign key
--                  in existing DsStDevice table record.
--
-- TABLES ACCESSED:  DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:  PARAMETER           DOMAIN
--          -----   -----
--          ServerId       serverid
--
-- OUTPUTS: PARAMETER           DOMAIN
--          -----   -----
--          NONE
--
-- ****
CREATE PROCEDURE DsStCPDelete
    @ServerId      serverid
AS
BEGIN

```

```

DECLARE @mystatus int
SELECT @mystatus = 0

-- Error checking to see if ServerId exists
IF NOT EXISTS(SELECT 1 FROM DsStConfigParameter WHERE
    ServerId = @ServerId)
BEGIN
    SELECT @mystatus = 1
    raiserror 32000 "DsStCPDelete: Unable to delete ServerId [%1!] as the server does not
exist.", @ServerId
    RETURN @mystatus
END

-- Error checking to see if Server is ON-LINE
IF EXISTS(SELECT 1 FROM DsStConfigParameter WHERE
    ServerId = @ServerId AND Status = "ON-LINE")
BEGIN
    SELECT @mystatus = 1
    raiserror 32001 "DsStCPDelete: Unable to Delete ServerId [%1!] as its current Status is
ON-LINE.", @ServerId
    RETURN @mystatus
END

-- Error checking to see if Device exists to Server
IF EXISTS (SELECT 1 FROM DsStDevice WHERE
    ServerId = @ServerId)
BEGIN
    SELECT @mystatus = 1
    raiserror 32002 "DsStCPDelete: Unable to Delete ServerId [%1!] as a Device Drive is
currently assigned to it. Either delete the Device or assign a different ServerId.", @ServerId
    RETURN @mystatus
END

BEGIN TRANSACTION

-- If the Servertype is a "STAGING MONITOR" we need to delete the
-- corresponding STAGING DISK entry.
IF (SELECT ServerType FROM DsStConfigParameter WHERE
    ServerId = @ServerId) = "STAGING MONITOR"
BEGIN
    DELETE DsStConfigParameter
    WHERE ServerId =
        substring (@ServerId, 1, (patindex("%Mon%", @ServerId) - 1))
        + "Dsk"
        + substring(@ServerId, (patindex("%Mon%", @ServerId) + 3), char_length(@ServerId))
END

SELECT @mystatus = @@error
IF (@mystatus !=0)
BEGIN
    SELECT @mystatus = 1

```

```

ROLLBACK TRANSACTION
raiserror 32003 "DsStCPDelete: Unable to delete information about ServerId [%1!]",  

@ServerId
    RETURN (@mystatus)
END

DELETE DsStConfigParameter
WHERE ServerId = @ServerId

SELECT @mystatus = @@error
IF (@mystatus !=0)
BEGIN
    SELECT @mystatus = 1
    ROLLBACK TRANSACTION
    raiserror 32004 "DsStCPDelete: Unable to delete information about ServerId [%1!]",  

@ServerId
    RETURN (@mystatus)
END

COMMIT TRANSACTION
RETURN (@mystatus)

END
go

```

## Procedure: DsStCPIInsertArchive

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStCPIInsertArchive
--
-- DESCRIPTION:      Inserts an Archive row into the DsStConfigParameter
--                   Table
--
-- TABLES ACCESSED:  DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS: PARAMETER           DOMAIN           SCREEN NAME
-- -----  -----
--       ServerId      serverid      Server ID
--       ServerType    servertype   Server Type
--       Rootpath      path        Root Path
--
-- OUTPUTS: PARAMETER           DOMAIN
-- -----  -----
-- 
```

```

--      NONE
-- ****
CREATE PROCEDURE DsStCPIInsertArchive
    @ServerId      serverid,
    @ServerType    servertype,
    @Rootpath      path
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    IF EXISTS (SELECT 1 FROM DsStConfigParameter WHERE
        ServerId = @ServerId)
        BEGIN
            SELECT @mystatus = 1
            raiserror 32005 "DsStCPIInsertArchive: The ServerId [%1!] you are attempting to insert
already exists.", @ServerId
            RETURN (@mystatus)
        END

        BEGIN TRANSACTION
        INSERT DsStConfigParameter(ServerId,
            ServerType,
            Status,
            Rootpath)
        values  (@ServerId,
            @ServerType,
            "OFF-LINE",
            @Rootpath)

        SELECT @mystatus = @@error
        IF (@mystatus != 0)
            BEGIN
                ROLLBACK TRANSACTION
                SELECT @mystatus = 1
                raiserror 32006 "DsStCPIInsertArchive: Unable to insert information about ServerId
[%1!].", @ServerId
                RETURN (@mystatus)
            END

            COMMIT TRANSACTION
            RETURN (@mystatus)
        END
    END
go

```

## Procedure: DsStCPIInsertDevice

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStCPIInsertDevice
--
-- DESCRIPTION:          Insert a resource/device server into the
--                       DsStConfigParameter table.
-- CREATE DATE:          9/18/97
-- CREATED BY:           Lisa Colombo
--
-- TABLES ACCESSED:     DsStConfigParameter
--
-- RETURNS               Status (Success = 0)
--
-- INPUT PARAMETERS      DOMAIN
-- -----
-- ServerId              serverid
-- ServerType             servertype
-- BlockSize              blocksize
-- Retries                smallcount
-- Sleeptime              sleeptime
-- LowCapacity            capacity
-- HighCapacity            capacity
-- TimeFactor              timefactor
-- DefaultBlockFactor      defaultblockfactor
-- TimerDuration            duration
-- TimerReload              duration
--
-- OUTPUT PARAMETERS      DOMAIN
-- -----
-- NONE
-- ****
CREATE PROCEDURE DsStCPIInsertDevice
    @ServerId      serverid,
    @ServerType     servertype,
    @BlockSize      blocksize,
    @Retries        smallcount,
    @Sleeptime      sleeptime,
    @LowCapacity    capacity,
    @HighCapacity   capacity,
    @TimeFactor     timefactor,
    @DefaultBlockFactor defaultblockfactor,
    @TimerDuration  duration,
    @TimerReload    duration
AS
BEGIN
    DECLARE @mystatus int
```

```

SELECT @mystatus = 0

IF EXISTS (SELECT 1 FROM DsStConfigParameter
           WHERE ServerId = @ServerId)
BEGIN
    SELECT @mystatus = 1
    raiserror 32007 "DsStCPIInsertDevice: The ServerId [%1!] you are attempting to
insert already exists.", @ServerId
    RETURN @mystatus
END

BEGIN TRANSACTION
INSERT DsStConfigParameter (
    ServerId,
    ServerType,
    Status,
    BlockSize,
    Retries,
    Sleepetime,
    LowCapacity,
    HighCapacity,
    TimeFactor,
    DefaultBlockFactor,
    TimerDuration,
    TimerReload)
VALUES (
    @ServerId,
    @ServerType,
    "OFF-LINE",
    @BlockSize,
    @Retries,
    @Sleepetime,
    @LowCapacity,
    @HighCapacity,
    @TimeFactor,
    @DefaultBlockFactor,
    @TimerDuration,
    @TimerReload)
SELECT @mystatus = @@error
IF (@mystatus !=0)
BEGIN
    SELECT @mystatus =1
    ROLLBACK TRANSACTION
    raiserror 32008 "DsStCPIInsertDevice: Unable to insert information about
ServerId [1%!].", @ServerId
    RETURN (@mystatus)
END

COMMIT TRANSACTION
RETURN (@mystatus)
END
go

```

## Procedure: DsStCPIInsertDisFtp

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStCPIInsertDisFtp
--
-- DESCRIPTION:      Inserts a row into the DsStConfigParameter Table
--
-- TABLES ACCESSED:  DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS: PARAMETER           DOMAIN           SCREEN NAME
-- -----
--   ServerId        serverid       Server ID
--   ServerType      servertype     Server Type
--   TotalSpace      space          Capacity
--   BlockSize       blocksize      Block Size
--   PullFtpHost    node           Pull FTP Host
--   PullFtpUserName username      Pull FTP User Name
--   PullFtpPassword password      Pull FTP Password
--   Datalist        datalist       Pull File Name
--   Retries         smallcount    Retries
--   Sleepetime      sleeptime    Sleepetime
--
-- OUTPUTS: PARAMETER           DOMAIN
-- -----
--   NONE
-- ****
CREATE PROCEDURE DsStCPIInsertDisFtp
    @ServerId      serverid,
    @ServerType     servertype,
    @TotalSpace     space,
    @BlockSize      blocksize,
    @PullFtpHost   node,
    @PullFtpUserName username,
    @PullFtpPassword password,
    @Datalist       datalist,
    @Retries        smallcount,
    @Sleepetime     sleeptime
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    IF EXISTS (SELECT 1 FROM DsStConfigParameter WHERE
               ServerId = @ServerId)
    BEGIN
```

```

SELECT @mystatus = 1
raiserror 32009 "DsStCPIinsertDisFtp: The ServerId [%1!] you are attempting to insert
already exists.", @ServerId
    RETURN (@mystatus)
END

BEGIN TRANSACTION
INSERT DsStConfigParameter(ServerId,
    ServerType,
    Status,
    TotalSpace,
    BlockSize,
    PullFtpHost,
    PullFtpUserName,
    PullFtpPassword,
    Datalist,
    Retries,
    Sleeptime)
values  (@ServerId,
    @ServerType,
    "OFF-LINE",
    @TotalSpace,
    @BlockSize,
    @PullFtpHost,
    @PullFtpUserName,
    @PullFtpPassword,
    @Datalist,
    @Retries,
    @Sleeptime)
SELECT @mystatus = @@error
IF (@mystatus != 0)
BEGIN
    SELECT @mystatus = 1
    ROLLBACK TRANSACTION
    raiserror 32010 "DsStCPIinsertDisFtp: Unable to insert information about ServerId [%1!].",
@ServerId
    RETURN (@mystatus)
END

COMMIT TRANSACTION
RETURN (@mystatus)
END
go

```

## **Procedure: DsStCPIinsertPullMonitor**

### **Code**

```

-- ****
-- BEGIN PROLOG
-- 
-- PROCEDURE NAME:    DsStCPIinsertPullMonitor

```

```

-- DESCRIPTION:      Inserts a Pull Monitor server type row.
-- DATE CREATED: 3/25/97
-- CREATED BY:      Marian Gowans
--
-- DATE MODIFIED:7/16/97 - LJC; Removed ZeroAccessFlag field.
-- DATE MODIFIED:8/5/97 - LJC; Restrict to 1 Pull Monitor insertion.
-- DATE MODIFIED:9/24/97 - LJC; Modify column names
-- DATE MODIFIED:10/24/97 - LJC; Modify columns
--
-- TABLES ACCESSED: DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS: PARAMETER           DOMAIN           SCREEN NAME
-- -----
--     ServerId        serverid       Server ID
--     ServerType       servertype    Server Type
--     OriginalCacheSpace   space
--     AvailableCacheSpace  space           Available Cache Space
--     BlockSize         blocksize      Block size
--     CacheReplacement   flag          Cache Replace
--     ConfirmDelete      boolean        Expired Files Confirm
--     MaxCommandLine     maxcommandline Max Command Line
--     ExpireNotify       boolean
--     PullExpirationTime timefactor    Expiration Time
--     MaxFiles          maxfiles      Max # of Files
--     PathLength         pathlength    Pathlen
--     HighWaterMark      watermark     Fault
--     LowWaterMark       watermark     Warning
--     Rootpath          path          Root Path
--     Datalist          datalist      Pull List
--     Directory          directory    Dir
--     FullPullMonitor    full
--     EC_KFTP_Command    path
-- -----
-- OUTPUTS: PARAMETER           DOMAIN
-- -----
--     NONE
-- ****

```

```

CREATE PROCEDURE DsStCPIInsertPullMonitor
    @ServerId        serverid,
    @ServerType       servertype,
    @OriginalCacheSpace   space,
    @AvailableCacheSpace  space,
    @BlockSize         blocksize,
    @CacheReplacement   flag,
    @ConfirmDelete      boolean,
    @MaxCommandLine     maxcommandline,
    @ExpireNotify       boolean,
    @PullExpirationTime timefactor,
    @MaxFiles          maxfiles,

```

```

@PathLength      pathlength,
@HighWaterMark   float,
@LowWaterMark    float,
@Rootpath        path,
@DataList         datalist,
@Directory       directory,
@FullPullMonitor full,
@EC_KFTP_Command path
AS
BEGIN
DECLARE @mystatus int
SELECT @mystatus = 0

IF EXISTS (SELECT 1 FROM DsStConfigParameter WHERE
           ServerType = "PULL MONITOR")
BEGIN
    SELECT @mystatus = 1
    raiserror 32011 "DsStCPIInsertPullMonitor: A Pull Monitor server already exists. Only
one is allowed.", @ServerId
    RETURN @mystatus
END

IF EXISTS (SELECT 1 FROM DsStConfigParameter WHERE
           ServerId = @ServerId)
BEGIN
    SELECT @mystatus = 1
    raiserror 32012 "DsStCPIInsertPullMonitor: The ServerId [%1!] you are attempting to
insert already exists.", @ServerId
    RETURN @mystatus
END

BEGIN TRANSACTION
INSERT DsStConfigParameter (ServerId,
                            ServerType,
                            Status,
                            OriginalCacheSpace,
                            AvailableCacheSpace,
                            BlockSize,
                            CacheReplacement,
                            ConfirmDelete,
                            MaxCommandLine,
                            ExpireNotify,
                            PullExpirationTime,
                            MaxFiles,
                            PathLength,
                            HighWaterMark,
                            LowWaterMark,
                            Rootpath,
                            Datalist,
                            Directory,
                            FullPullMonitor,
                            EC_KFTP_Command)

```

```

values      (@ServerId,
@ServerType,
"OFF-LINE",
@OriginalCacheSpace,
@AvailableCacheSpace,
@BlockSize,
@CacheReplacement,
@ConfirmDelete,
@MaxCommandLine,
          @ExpireNotify,
@PullExpirationTime,
@MaxFiles,
@PathLength,
@HighWaterMark,
@LowWaterMark,
@Rootpath,
@Datalist,
@Directory,
          @FullPullMonitor,
          @EC_KFTP_Command)
SELECT @mystatus = @@error
IF (@mystatus !=0)
BEGIN
    SELECT @mystatus = 1
    ROLLBACK TRANSACTION
    raiserror 32013 "DsStCPIInsertPullMonitor: Unable to insert information about ServerId
[%1!].", @ServerId
    RETURN (@mystatus)
END
COMMIT TRANSACTION
RETURN (@mystatus)
END
go

```

## **Procedure: DsStCPIInsertStagingMonitor**

### **Code**

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStCPIInsertStagingMonitor
--
-- DESCRIPTION:      Inserts a Staging Monitor row and a Staging Disk row
-- DATE CREATED: 3/25/97
--
-- DATE MODIFIED:7/16/97; LJC - Removed ZeroAccessFlag field from Database
-- DATE MODIFIED:10/27/97; LJC - Corrected naming of StagingDisk Server
--
-- TABLES ACCESSED:  DsStConfigParameter
-- 
```

```

-- RETURNS:      Status (Success = 0)
--
-- INPUTS: PARAMETER          DOMAIN           SCREEN NAME
-----
--   ServerId      serverid      Server ID
--   ServerType     servertype    Server Type
--   TotalSpace     space        Total Space
--   CacheSpace     space        Cache Space
--   UserStagingSpace space       User Staging
--   HighWaterMark  watermark   Fault/High Mark
--   LowWaterMark   watermark   Warning/Low Mark
--   BlockSize      blocksize   Block Size
--   MaxCommandLine maxcommandline Max Command Line
--   MaxFiles       maxfiles    Max # Files Cache
--   PathLength     path        Pathlen
--   Rootpath       path        Root Path
--   Datalist       datalist    Data List
--
-- OUTPUTS: PARAMETER          DOMAIN
-----
--   NONE
-- ****
CREATE PROCEDURE DsStCPIInsertStagingMonitor
    @ServerId      serverid,
    @ServerType     servertype,
    @TotalSpace     space,
    @CacheSpace     space,
    @UserStagingSpace space,
    @HighWaterMark  float,
    @LowWaterMark   float,
    @BlockSize      blocksize,
    @MaxCommandLine maxcommandline,
    @MaxFiles       maxfiles,
    @PathLength     pathlength,
    @Rootpath       path,
    @Datalist       datalist
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    IF EXISTS (SELECT 1 FROM DsStConfigParameter WHERE
               ServerId = @ServerId)
        BEGIN
            SELECT @mystatus = 1
            raiserror 32014 "DsStCPIInsertStagingMonitor: The ServerId [%1!] you are attempting to
            insert already exists.", @ServerId
            RETURN (@mystatus)
        END

    BEGIN TRANSACTION
    INSERT DsStConfigParameter(ServerId,

```

```

        ServerType,
        Status,
        TotalSpace,
        CacheSpace,
        UserStagingSpace,
        HighWaterMark,
        LowWaterMark,
        BlockSize,
        MaxCommandLine,
        MaxFiles,
        PathLength,
        Rootpath,
        Datalist)
values  (@ServerId,
        @ServerType,
        "OFF-LINE",
        @TotalSpace,
        @CacheSpace,
        @UserStagingSpace,
        @HighWaterMark,
        @LowWaterMark,
        @BlockSize,
        @MaxCommandLine,
        @MaxFiles,
        @PathLength,
        @Rootpath,
        @Datalist)

SELECT @mystatus = @@error
IF (@mystatus != 0)
BEGIN
    ROLLBACK TRANSACTION
    SELECT @mystatus = 1
    raiserror 32015 "DsStCPIInsertStagingMonitor: Unable to insert information about Staging
Monitor ServerId [%1!].", @ServerId
    RETURN (@mystatus)
END

INSERT DsStConfigParameter(ServerId,
                           ServerType,
                           Status,
                           TotalSpace,
                           CacheSpace,
                           UserStagingSpace,
                           Rootpath)
values  (substring(@ServerId,
                   1,
                   (patindex("%Monitor%", @ServerId) - 1))
           + "Disk"
           + substring(@ServerId,
                       (patindex("%Monitor%", @ServerId) + 7),
                       char_length(@ServerId)),

```

```

    "STAGING DISK",
    "OFF-LINE",
    @TotalSpace,
    @CacheSpace,
    @UserStagingSpace,
    @Rootpath)

SELECT @mystatus = @@error
IF (@mystatus != 0)
BEGIN
    ROLLBACK TRANSACTION
    SELECT @mystatus = 1
    raiserror 32016 "DsStCPIInsertStagingMonitor: Unable to insert information about Staging
Disk ServerId [%1!].", @ServerId
    RETURN (@mystatus)
END
COMMIT TRANSACTION
RETURN (@mystatus)

END
go

```

## **Procedure: DsStCPSelectById**

### **Code**

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:   DsStCPSelectById
--
-- DESCRIPTION:      Selects a row from the DsStConfigParameter Table
-- DATE CREATED: 3/25/97
--
-- DATE MODIFIED:7/16/97; LJC - Removed ZeroAccessFlag from Database
-- DATE MODIFIED:9/26/97; LJC - Modified columns in Database
-- DATE MODIFIED:   10/24/97; LJC - modified columns
--
-- TABLES ACCESSED: DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:  PARAMETER        DOMAIN
--          -----  -----
--          ServerId       serverid
--
-- OUTPUTS: PARAMETER        DOMAIN
--          -----  -----
--          ServerId       serverid
--          Status         state
--          Node          node
--          TotalSpace     space

```

```

--      CacheSpace      space
--      UserStagingSpace space
--      OriginalCacheSpace space
--      AvailableCacheSpace space
--      LowWaterMark      watermark
--      Rootpath         path
--      HighWaterMark    watermark
--      ConfirmDelete    boolean
--      MaxFiles        maxfiles
--          ExpireNotify   boolean
--          PullExpirationTime timefactor
--      BlockSize        blocksize
--      PathLength       pathlength
--      Datalist         datalist
--      MaxCommandLine   maxcommandline
--      Directory        directory
--      PullFtpHost      node
--      PullFtpUserName  username
--      PullFtpPassword  password
--      Retries          smallcount
--      Sleepetime       sleepetime
--      CacheId          cacheid
--      ServerType       servertype
--      CacheReplacement flag
--          FullPullMonitor full
--      EC_KFTP_Command  path
--      NumColumns       int
--      NumRows          int
--      PackingSlips     smallcount
--      PrintQue         file
-- *****

CREATE PROCEDURE DsStCPSelectById
    @ServerId      serverid
AS
BEGIN
    DECLARE @mystatus int

    SELECT @mystatus = 0

    -- Error checking to see if ServerId requested actually exists
    IF NOT EXISTS(SELECT 1 FROM DsStConfigParameter WHERE
                  ServerId = @ServerId)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32017 "DsStCPSelectById: ServerId [% 1!] does not exist.", @ServerId
        RETURN (@mystatus)
    END

    SELECT "ServerId",           ServerId,
          "Status",            Status,
          "Node",              Node,

```

```

"TotalSpace",           TotalSpace,
"CacheSpace",          CacheSpace,
"UserStagingSpace",    UserStagingSpace,
"OriginalCacheSpace", OriginalCacheSpace,
"AvailableCacheSpace", AvailableCacheSpace,
"LowWaterMark",        LowWaterMark,
"HighWaterMark",       HighWaterMark,
"Rootpath",            Rootpath,
"ConfirmDelete",       ConfirmDelete,
"ExpireNotify",        ExpireNotify,
"PullExpirationTime", PullExpirationTime,
"MaxFiles",            MaxFiles ,
"BlockSize",           BlockSize,
"PathLength",          PathLength,
"Datalist",            Datalist,
"MaxCommandLine",     MaxCommandLine,
"Directory",           Directory,
"PullFtpHost",          PullFtpHost,
"PullFtpUserName",     PullFtpUserName,
"PullFtpPassword",      PullFtpPassword,
"Retries",              Retries,
"Sleepetime",           Sleepetime,
"CacheId",              CacheId,
"ServerType",           ServerType,
"CacheReplacement",    convert(integer, CacheReplacement),
"FullPullMonitor",     FullPullMonitor,
"EC_KFTP_Command",     EC_KFTP_Command,
"NumColumns",           NumColumns,
"NumRows",              NumRows,
"PackingSlips",         PackingSlips,
"PrintQue",             PrintQue
FROM DsStConfigParameter
WHERE ServerId = @ServerId

SELECT @mystatus = @@error
IF (@mystatus !=0)
BEGIN
    SELECT @mystatus = 1
    raiserror 32018 "DsStCPSelectById: Unable to select information about ServerId [%1!]", 
    @ServerId
    RETURN (@mystatus)
END
RETURN (@mystatus)
END
go

```

## Procedure: DsStCPSelectByType

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStCPSelectByType
--
-- DESCRIPTION:      Selects rows from the DsStConfigParameter Table
--                   based on ServerType
-- DATE CREATED: 3/25/97
--
-- MODIFICATION: 7/16/97; LJC; Removed ZeroAccessFlag from Database
--                 9/10/97; LJC; Eliminated Unused Outputs
--
-- TABLES ACCESSED: DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER          DOMAIN
--                   -----
--                   ServerType        servertype
--
-- OUTPUTS:          PARAMETER          DOMAIN
--                   -----
--                   ServerId         serverid
--                   Status           state
--                   Node             node
--
-- ****
CREATE PROCEDURE DsStCPSelectByType
    @ServerType        servertype
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0
    --Error checking to see if ServerType requested actually exists
    IF NOT EXISTS(SELECT 1 FROM DsStServerType WHERE
                  ServerType = @ServerType)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32019 "DsStCPSelectByType: ServerType [%1!] does not exist.", @ServerType
        RETURN (@mystatus)
    END
    SELECT "ServerId",           ServerId,
          "Status",            Status,
          "Node",              Node
    FROM   DsStConfigParameter
```

```

WHERE ServerType      = @ServerType

SELECT @mystatus = @@error
IF (@mystatus !=0)
BEGIN
    SELECT @mystatus = 1
    raiserror 32020 "DsStCPSelectByType: Unable to select information about ServerType
[%1!]", @ServerType
    RETURN (@mystatus)
END
RETURN (@mystatus)
END
go

```

## Procedure: DsStCPUpdate

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStCPUpdate
--
-- DESCRIPTION:          Updates any server row.
-- CREATE DATE:          10/20/97
-- CREATED BY:           Padmaja Akkineni
--
-- DATE MODIFIED:10/28/97; LJC - modified column names
--
-- TABLES ACCESSED:     DsStConfigParameter
--
-- RETURNS:              Status (Success = 0)
--
-- INPUT PARAMETERS        DOMAIN
-- -----
-- ServerId                serverid
-- ServerType               servertype
-- BlockSize                blocksize
-- Retries                  smallcount
-- Sleeptime                sleeptime
-- LowCapacity              capacity
-- HighCapacity             capacity
-- TimeFactor               timefactor
-- DefaultBlockFactor       defaultblockfactor
-- TimerDuration            duration
-- TimerReload               duration
-- TotalSpace                space
-- PullFtpHost              node
-- PullFtpUserName          username
-- PullFtpPassword           password
-- Datalist                  datalist
-- AvailableCacheSpace       space

```

```

-- CacheReplacement      flag
-- ConfirmDelete        boolean
-- MaxCommandLine       maxcommandline
-- ExpireNotify         boolean
-- PullExpirationTime   timefactor
-- MaxFiles              maxfiles
-- PathLength            pathlength
-- HighWaterMark         float
-- LowWaterMark          float
-- Directory             directory
-- FullPullMonitor       full
-- Rootpath              path
-- EC_KFTP_Command      path
-- NumColumns            int
-- NumRows               int
-- PackingSlips          smallcount
-- PrintQue              file
-- 

-- OUTPUT PARAMETERS      DOMAIN
----- -----
-- NONE
-- ****
CREATE PROCEDURE DsStCPUUpdate
    @ServerId           serverid,
    @ServerType          servertype,
    @BlockSize           blocksize,
    @Retries              smallcount,
    @Sleepetime          sleeptime,
    @LowCapacity          capacity,
    @HighCapacity         capacity,
    @TimeFactor           timefactor,
    @DefaultBlockFactor   defaultblockfactor,
    @TimerDuration        duration,
    @TimerReload          duration,
    @TotalSpace           space,
    @PullFtpHost          node,
    @PullFtpUserName      username,
    @PullFtpPassword      password,
    @Datalist              datalist,
    @AvailableCacheSpace   space,
    @CacheReplacement      flag,
    @ConfirmDelete         boolean,
    @MaxCommandLine        maxcommandline,
    @ExpireNotify          boolean,
    @PullExpirationTime    timefactor,
    @MaxFiles              maxfiles,
    @PathLength            pathlength,
    @HighWaterMark          float,
    @LowWaterMark           float,
    @Directory             directory,
    @FullPullMonitor       full,
    @Rootpath              path,

```

```

        @EC_KFTP_Command      path,
@NumColumns          int,
@NumRows              int,
@PackingSlips         smallcount,
@PrintQue             file

AS
BEGIN
    DECLARE @mystatus           int

-- Temporary variables to hold the values from the table

    DECLARE @serverType          servertype
    DECLARE @blockSize            blocksize
    DECLARE @retries              smallcount
    DECLARE @sleeptime            sleeptime
    DECLARE @lowCapacity          capacity
    DECLARE @highCapacity          capacity
    DECLARE @timeFactor           timefactor
    DECLARE @defaultBlockFactor   defaultblockfactor
    DECLARE @timerDuration        duration
    DECLARE @timerReload          duration
    DECLARE @totalSpace            space
    DECLARE @pullFtpHost          node
    DECLARE @pullFtpUserName       username
    DECLARE @pullFtpPassword       password
    DECLARE @datalist              datalist
    DECLARE @availableCacheSpace   space
    DECLARE @cacheReplacement      flag
    DECLARE @confirmDelete         boolean
    DECLARE @maxCommandLine        maxcommandline
    DECLARE @expireNotify          boolean
    DECLARE @pullExpirationTime    timefactor
    DECLARE @maxFiles              maxfiles
    DECLARE @pathLength            pathlength
    DECLARE @highWaterMark         float
    DECLARE @lowWaterMark          float
    DECLARE @directory              directory
    DECLARE @fullPullMonitor       full
    DECLARE @rootpath               path
    DECLARE @ec_KFTP_Command       path
    DECLARE @numColumns            int
    DECLARE @numRows                int
    DECLARE @packingSlips          smallcount
    DECLARE @printQue               file

    SELECT @mystatus = 0

-- Error Checking to see if ServerId exists.
    IF NOT EXISTS (SELECT 1 FROM DsStConfigParameter
                  WHERE ServerId = @ServerId)

```

```

BEGIN
    SELECT @mystatus = 1
    raiserror 32033 "DsStCPUpdate: The device server you are attempting to update
[%1!] does not exist.", @ServerId

        RETURN(@mystatus)
END

SELECT @serverType      = ServerType,
       @blockSize       = BlockSize,
       @retries         = Retries,
       @sleepetime     = Sleepetime,
       @lowCapacity    = LowCapacity,
       @highCapacity   = HighCapacity,
       @timeFactor     = TimeFactor,
       @defaultBlockFactor = DefaultBlockFactor,
       @timerDuration  = TimerDuration,
       @timerReload    = TimerReload,
       @totalSpace     = TotalSpace,
       @pullFtpHost   = PullFtpHost,
       @pullFtpUserName = PullFtpUserName,
       @pullFtpPassword = PullFtpPassword,
       @datalist       = Datalist,
       @availableCacheSpace = AvailableCacheSpace,
       @cacheReplacement = CacheReplacement,
       @confirmDelete   = ConfirmDelete,
       @maxCommandLine = MaxCommandLine,
       @expireNotify    = ExpireNotify,
       @pullExpirationTime = PullExpirationTime,
       @maxFiles        = MaxFiles,
       @pathLength      = PathLength,
       @highWaterMark   = HighWaterMark,
       @lowWaterMark    = LowWaterMark,
       @directory       = Directory,
       @fullPullMonitor = FullPullMonitor,
       @rootpath        = Rootpath,
       @ec_KFTP_Command = EC_KFTP_Command,
       @numColumns      = NumColumns,
       @ numRows         = NumRows,
       @packingSlips    = PackingSlips,
       @printQue        = PrintQue
FROM DsStConfigParameter
WHERE ServerId      = @ServerId

-- TO check if the parameter has some value and if so,
-- replaces with the temporaray variable with parameter

IF ( @ServerType != NULL )
    SELECT @serverType = @ServerType
IF ( @BlockSize != 0 )
    SELECT @blockSize = @BlockSize
IF ( @Retries != 0 )

```

```

        SELECT @retries = @Retries
IF ( @SleepTime != 0 )
    SELECT @sleepTime = @SleepTime
IF ( @LowCapacity != 0 )
    SELECT @lowCapacity = @LowCapacity
IF ( @HighCapacity != 0 )
    SELECT @highCapacity = @HighCapacity
IF ( @TimeFactor != 0 )
    SELECT @TimeFactor = @TimeFactor
IF ( @DefaultBlockFactor != 0 )
    SELECT @defaultBlockFactor = @DefaultBlockFactor
IF ( @TimerDuration != 0 )
    SELECT @timerDuration = @TimerDuration
IF ( @TimerReload != 0 )
    SELECT @timerReload = @TimerReload
IF ( @TotalSpace != 0 )
    SELECT @totalSpace = @TotalSpace
IF ( @PullFtpHost != NULL )
    SELECT @pullFtpHost = @PullFtpHost
IF ( @PullFtpUserName != NULL )
    SELECT @pullFtpUserName = @PullFtpUserName
IF ( @PullFtpPassword != NULL )
    SELECT @pullFtpPassword = @PullFtpPassword
IF ( @Datalist != NULL )
    SELECT @Datalist = @Datalist
IF ( @AvailableCacheSpace != 0 )
    SELECT @availableCacheSpace = @AvailableCacheSpace
IF ( @CacheReplacement != NULL )
    SELECT @cacheReplacement = @CacheReplacement
IF ( @ConfirmDelete != NULL )
    SELECT @confirmDelete = @ConfirmDelete
IF ( @MaxCommandLine != 0 )
    SELECT @maxCommandLine = @MaxCommandLine
IF ( @ExpireNotify != NULL )
    SELECT @expireNotify = @ExpireNotify
IF ( @PullExpirationTime != 0 )
    SELECT @pullExpirationTime = @PullExpirationTime
IF ( @MaxFiles != 0 )
    SELECT @maxFiles = @MaxFiles
IF ( @PathLength != 0 )
    SELECT @pathLength = @PathLength
IF ( @HighWaterMark != 0 )
    SELECT @highWaterMark = @HighWaterMark
IF ( @LowWaterMark != 0 )
    SELECT @lowWaterMark = @LowWaterMark
IF ( @Directory != NULL )
    SELECT @directory = @Directory
IF ( @FullPullMonitor != 0 )
    SELECT @fullPullMonitor = @FullPullMonitor
IF ( @Rootpath != NULL )
    SELECT @rootpath = @Rootpath
IF ( @EC_KFTP_Command != NULL )

```

```

        SELECT @EC_KFTP_Command = @ec_KFTP_Command
IF ( @NumColumns != NULL )
    SELECT @numColumns = @NumColumns
IF ( @NumRows != NULL )
    SELECT @numRows = @NumRows
IF ( @PackingSlips != NULL )
    SELECT @packingSlips = @PackingSlips
IF ( @PrintQue != NULL )
    SELECT @printQue = @PrintQue

BEGIN TRANSACTION
UPDATE DsStConfigParameter
SET ServerId          = @ServerId,
    ServerType        = @serverType,
    BlockSize         = @blockSize,
    Retries           = @retries,
    Sleepetime        = @sleepetime,
    LowCapacity       = @lowCapacity,
    HighCapacity      = @highCapacity,
    TimeFactor        = @timeFactor,
    DefaultBlockFactor= @DefaultBlockFactor,
    TimerDuration     = @timerDuration,
    TimerReload       = @timerReload,
    TotalSpace        = @totalSpace,
    PullFtpHost       = @pullFtpHost,
    PullFtpUserName   = @pullFtpUserName,
    PullFtpPassword   = @pullFtpPassword,
    Datalist          = @datalist,
    AvailableCacheSpace= @availableCacheSpace,
    CacheReplacement  = @cacheReplacement,
    ConfirmDelete     = @confirmDelete,
    MaxCommandLine    = @maxCommandLine,
    ExpireNotify      = @expireNotify,
    PullExpirationTime= @pullExpirationTime,
    MaxFiles          = @maxFiles,
    PathLength         = @pathLength,
    HighWaterMark     = @highWaterMark,
    LowWaterMark       = @lowWaterMark,
    Directory          = @directory,
    FullPullMonitor   = @fullPullMonitor,
    Rootpath           = @rootpath,
    EC_KFTP_Command   = @ec_KFTP_Command,
    NumColumns         = @numColumns,
    NumRows            = @numRows,
    PackingSlips       = @packingSlips,
    PrintQue          = @printQue

```

WHERE ServerId = @ServerId

SELECT @mystatus = @@error

IF (@mystatus != 0)

```

BEGIN
    SELECT @mystatus = 1
    ROLLBACK TRANSACTION
    raiserror 32034 "DsStCPUpdate: Unable to update information for (device)
Server [%1!].", @ServerId
    END
ELSE
    BEGIN
        COMMIT TRANSACTION
        RETURN(@mystatus)
    END
END
go

```

## Procedure: DsStCPUpdateArchive

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStCPUpdateArchive
--
-- DESCRIPTION:      Update a row into the DsStConfigParameter Table
--
-- TABLES ACCESSED:  DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS: PARAMETER           DOMAIN           SCREEN NAME
-- -----
--     ServerId       serverid      Server ID
--     ServerType     servertype   Server Type
--     Rootpath       path         Root Path
--
-- OUTPUTS: PARAMETER           DOMAIN
-- -----
--     NONE
-- ****
CREATE PROCEDURE DsStCPUpdateArchive
    @ServerId      serverid,
    @ServerType    servertype,
    @Rootpath      path
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    IF NOT EXISTS (SELECT 1 FROM DsStConfigParameter WHERE
                    ServerId = @ServerId)
    BEGIN
        SELECT @mystatus = 1
    END

```

```

raiserror 32027 "DsStCPUpdateArchive: Unable to update ServerId [%1!] as ServerId does
not currently exist.", @ServerId
    RETURN (@mystatus)
END

BEGIN TRANSACTION
UPDATE DsStConfigParameter
    SET Rootpath = @Rootpath
    WHERE ServerId = @ServerId

SELECT @mystatus = @@error
IF (@mystatus != 0)
BEGIN
    ROLLBACK TRANSACTION
    SELECT @mystatus = 1
    raiserror 32028 "DsStCPUpdateArchive: Unable to update information about ServerId
[%1!].", @ServerId
    RETURN (@mystatus)
END

COMMIT TRANSACTION
RETURN (@mystatus)

END
go

```

## Procedure: DsStCPUpdateDevice

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStCPUpdateDevice
--
-- DESCRIPTION:         Updates a (device) server row.
-- CREATE DATE:        09/22/97
-- CREATED BY:         Lisa Colombo
--
-- TABLES ACCESSED:    DsStConfigParameter
--
-- RETURNS:             Status (Success = 0)
--
-- INPUT PARAMETERS          DOMAIN
-- -----
-- ServerId                serverid
-- ServerType               servertype
-- BlockSize                blocksize
-- Retries                  smallcount
-- Sleeptime                sleeptime
-- LowCapacity              capacity
-- HighCapacity             capacity

```

```

-- TimeFactor          timefactor
-- DefaultBlockFactor defaultblockfactor
-- TimerDuration      duration
-- TimerReload         duration
--
-- OUTPUT PARAMETERS    DOMAIN
-----
-- NONE
-- ****
CREATE PROCEDURE DsStCPUpdateDevice
    @ServerId           serverid,
    @ServerType          servertype,
    @BlockSize           blocksize,
    @Retries             smallcount,
    @Sleepetime          sleeptime,
    @LowCapacity         capacity,
    @HighCapacity        capacity,
    @TimeFactor          timefactor,
    @DefaultBlockFactor defaultblockfactor,
    @TimerDuration       duration,
    @TimerReload         duration
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0
    -- Error Checking to see if ServerId exists.
    IF NOT EXISTS (SELECT 1 FROM DsStConfigParameter
                   WHERE ServerId = @ServerId)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32029 "DsStCPUpdateDevice: The device server you are attempting to
update [%1!] does not exist.", @ServerId
        RETURN(@mystatus)
    END
    BEGIN TRANSACTION
    UPDATE DsStConfigParameter
    SET   ServerId          = @ServerId,
          ServerType         = @ServerType,
          BlockSize          = @BlockSize,
          Retries            = @Retries,
          Sleepetime         = @Sleepetime,
          LowCapacity        = @LowCapacity,
          HighCapacity       = @HighCapacity,
          TimeFactor         = @TimeFactor,
          DefaultBlockFactor = @DefaultBlockFactor,
          TimerDuration      = @TimerDuration,
          TimerReload        = @TimerReload

```

```

WHERE ServerId = @ServerId

SELECT @mystatus = @@error
IF (@mystatus != 0)
BEGIN
    SELECT @mystatus = 1
    ROLLBACK TRANSACTION
    raiserror 32030 "DsStCPUpdateDevice: Unable to update information for
(device) Server [%1!].", @ServerId
    END
ELSE
BEGIN
    COMMIT TRANSACTION
    RETURN(@mystatus)
END
END
go

```

## Procedure: DsStCPUpdateDisFtp

### Code

```

-- *****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStCPUpdateDisFtp
--
-- DESCRIPTION:      Updates a row into the DsStConfigParameter Table
--
-- TABLES ACCESSED:  DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS: PARAMETER           DOMAIN           SCREEN NAME
-- -----
--     ServerId        serverid        Server ID
--     ServerType       servertype      Server Type
--     TotalSpace       space          Capacity
--     BlockSize        blocksize      Block size
--     PullFtpHost      node          Pull FTP Host
--     PullFtpUserName  username      Pull FTP User Name
--     PullFtpPassword   password      Pull FTP Password
--     Datalist         datalist      Pull File Name
--     Retries          smallcount    Retries
--     Sleeptime        sleeptime    Sleeptime
--
-- OUTPUTS: PARAMETER           DOMAIN
-- -----
--     NONE
-- *****
CREATE PROCEDURE DsStCPUpdateDisFtp
    @ServerId        serverid,

```

```

@ServerType    servertype,
@TotalSpace    space,
@BlockSize     blocksize,
@PullFtpHost   node,
@PullFtpUserName username,
@PullFtpPassword password,
@DataList       datalist,
@Retries        smallcount,
@SleepTime      sleeptime
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    IF NOT EXISTS (SELECT 1 FROM DsStConfigParameter WHERE
                    ServerId = @ServerId)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32031 "DsStCPUpdateDisFtp: The ServerId [%1!] you are attempting to update
does not exist.", @ServerId
        RETURN (@mystatus)
    END

    BEGIN TRANSACTION
    UPDATE DsStConfigParameter
        SET TotalSpace    = @TotalSpace,
            BlockSize     = @BlockSize,
            PullFtpHost   = @PullFtpHost,
            PullFtpUserName = @PullFtpUserName,
            PullFtpPassword = @PullFtpPassword,
            DataList       = @DataList,
            Retries        = @Retries,
            SleepTime      = @SleepTime
        WHERE ServerId    = @ServerId

        SELECT @mystatus = @@error
        IF (@mystatus != 0)
        BEGIN
            SELECT @mystatus = 1
            ROLLBACK TRANSACTION
            raiserror 32032 "DsStCPUpdateDisFtp: Unable to update information about ServerId
[%1!].", @ServerId
            RETURN (@mystatus)
        END

        COMMIT TRANSACTION
        RETURN (@mystatus)
    END
go

```

## Procedure: DsStCPUpdatePullMonitor

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStCPUpdatePullMonitor
--
-- DECSRIPTION:      Updates a Pull Monitor server row.
-- DATE CREATED: 3/25/97
--
-- DATE MODIFIED:7/16/97; LJC - removed ZeroAccessFlag from Database
-- DATE MODIFIED:9/24/97; LJC - modified column names
-- DATE MODIFIED:10/24/97; LJC - modified columns
--
-- TABLES ACCESSED:  DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS: PARAMETER           DOMAIN             SCREEN NAME
-- -----
--     ServerId        serverid       Server ID
--     ServerType       servertype     Server Type
--     OriginalCacheSpace   space
--     AvailableCacheSpace   space      Available Cache Space
--     BlockSize         blocksize     Block size
--     CacheReplacement    flag        Cache Replace
--     ConfirmDelete      boolean      Expired Files Confirm
--     MaxCommandLine     maxcommandline Max Command Line
--     ExpireNotify       boolean
--     PullExpirationTime timefactor   Expiration Time
--     MaxFiles          maxfiles     Max # of Files
--     PathLength         pathlength   Pathlen
--     HighWaterMark      watermark   Fault
--     LowWaterMark       watermark   Warning
--     Rootpath          path        Root Path
--     Datalist          datalist     Pull List
--     Directory          directory   Dir
--     FullPullMonitor    full
--     EC_KFTP_Command    path
--
-- OUTPUTS: PARAMETER           DOMAIN
-- -----
--     NONE
-- ****
CREATE PROCEDURE DsStCPUpdatePullMonitor
    @ServerId        serverid,
    @ServerType       servertype,
    @AvailableCacheSpace   space,
    @BlockSize        blocksize,
```

```

@CacheReplacement      flag,
@ConfirmDelete         boolean,
@MaxCommandLine        maxcommandline,
@ExpireNotify          boolean,
@PullExpirationTime   timefactor,
@MaxFiles               maxfiles,
@PathLength              pathlength,
@HighWaterMark           float,
@LowWaterMark            float,
@Rootpath                path,
@Datalist                 datalist,
@Directory                directory,
@FullPullMonitor          full,
@EC_KFTP_Command        path
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    IF NOT EXISTS (SELECT 1 FROM DsStConfigParameter WHERE
                    ServerId = @ServerId)
        BEGIN
            SELECT @mystatus = 1
            raiserror 32035 "DsStCPUpdatePullMonitor: The Pull Monitor Server you are attempting to
update [% !] does not exist.", @ServerId
            RETURN @mystatus
        END

    BEGIN TRANSACTION
    UPDATE DsStConfigParameter
        SET AvailableCacheSpace = @AvailableCacheSpace,
            BlockSize = @BlockSize,
            CacheReplacement = @CacheReplacement,
            ConfirmDelete = @ConfirmDelete,
            MaxCommandLine = @MaxCommandLine,
            ExpireNotify = @ExpireNotify,
            PullExpirationTime = @PullExpirationTime,
            MaxFiles = @MaxFiles,
            PathLength = @PathLength,
            HighWaterMark = @HighWaterMark,
            LowWaterMark = @LowWaterMark,
            Rootpath = @Rootpath,
            Datalist = @Datalist,
            Directory = @Directory,
            FullPullMonitor = @FullPullMonitor,
            EC_KFTP_Command = @EC_KFTP_Command
    WHERE ServerId = @ServerId

    SELECT @mystatus = @@error
    IF (@mystatus !=0)
        BEGIN
            SELECT @mystatus = 1
        END

```

```

ROLLBACK TRANSACTION
raiserror 32036 "DsStCPUpdatePullMonitor: Unable to update information for Pull
Monitor Server [%1!].", @ServerId
    RETURN (@mystatus)
END

COMMIT TRANSACTION
RETURN (@mystatus)
END
go

```

## Procedure: DsStCPUpdateStagingMonitor

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStCPUpdateStagingMonitor
--
-- DESCRIPTION:      Updates a Staging Monitor row.
-- DATE CREATED: 3/25/97
--
-- DATE MODIFIED:7/16/97; LJC - Removed ZeroAccessFlag from Database.
--
-- TABLES ACCESSED:  DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS: PARAMETER           DOMAIN           SCREEN NAME
-- -----
--     ServerId        serverid       Server ID
--     ServerType      servertype     Server Type
--     TotalSpace      space         Total Space
--     CacheSpace      space         Cache Space
--     UserStagingSpace space         User Staging
--     BlockSize       blocksize     Block Size
--     MaxCommandLine maxcommandline Max Command Line
--     MaxFiles        maxfiles      Max # Files Cache
--     PathLength      pathlength    Pathlen
--     Rootpath        path         Root Path
--     Datalist        datalist      Data List
--
-- OUTPUTS: PARAMETER           DOMAIN
-- -----
--     NONE
-- ****
CREATE PROCEDURE DsStCPUpdateStagingMonitor
    @ServerId      serverid,
    @ServerType     servertype,
    @TotalSpace     space,
    @CacheSpace     space,

```

```

@UserStagingSpace space,
@BlockSize      blocksize,
@MaxCommandLine maxcommandline,
@MaxFiles       maxfiles,
@PathLength     pathlength,
@Rootpath       path,
@Datalist       datalist
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    IF NOT EXISTS (SELECT 1 FROM DsStConfigParameter WHERE
        ServerId = @ServerId)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32037 "DsStCPUpdateStagingMonitor: The ServerId [%1!] you are attempting to
update does not exist.", @ServerId
        RETURN (@mystatus)
    END

    BEGIN TRANSACTION
    UPDATE DsStConfigParameter
        SET TotalSpace      = @TotalSpace,
            CacheSpace      = @CacheSpace,
            UserStagingSpace = @UserStagingSpace,
            BlockSize       = @BlockSize,
            MaxCommandLine  = @MaxCommandLine,
            MaxFiles        = @MaxFiles,
            PathLength      = @PathLength,
            Rootpath        = @Rootpath,
            Datalist        = @Datalist
        WHERE ServerId      = @ServerId

        SELECT @mystatus = @@error
        IF (@mystatus != 0)
        BEGIN
            ROLLBACK TRANSACTION
            SELECT @mystatus = 1
            raiserror 32038 "DsStCPUpdateStagingMonitor: Unable to update information about
ServerId [%1!].", @ServerId
            RETURN (@mystatus)
        END

        UPDATE DsStConfigParameter
            SET TotalSpace      = @TotalSpace,
                CacheSpace      = @CacheSpace,
                UserStagingSpace = @UserStagingSpace,
                Rootpath        = @Rootpath
        WHERE ServerId =
            substring (@ServerId, 1, (patindex("%Mon%", @ServerId) - 1))
            + "Dsk"

```

```

+ substring(@ServerId, (patindex("%Mon%", @ServerId) + 3), char_length(@ServerId))

SELECT @mystatus = @@error
IF (@mystatus != 0)
BEGIN
    ROLLBACK TRANSACTION
    SELECT @mystatus = 1
    raiserror 32039 "DsStCPUUpdateStagingMonitor: Unable to update information about
ServerId [%1!].", @ServerId
    RETURN (@mystatus)
END

COMMIT TRANSACTION
RETURN (@mystatus)

END
go

```

## Procedure: DsStCPUUpdAvailableCacheSpace

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStCPUUpdAvailableCacheSpace
--
-- DESCRIPTION: Update the AvailableCacheSpace field for the Pull
--               Monitor Server.
--
-- TABLES ACCESSED:   DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER      DOMAIN
--                   -----      -----
--                   ServerId      serverid
--                   Amount        int
--
-- OUTPUTS:          PARAMETER      DOMAIN
--                   -----      -----
--                   NONE
-- ****
CREATE PROCEDURE DsStCPUUpdAvailableCacheSpace
    @ServerId      serverid,
    @Amount        int
AS
BEGIN
    DECLARE @mystatus int
    DECLARE @AvailableCacheSpace int

    SELECT @AvailableCacheSpace = 0

```

```

SELECT @mystatus = 0

IF NOT EXISTS (SELECT 1 FROM DsStConfigParameter WHERE
    @ServerId = ServerId)
BEGIN
    SELECT @mystatus = 1
    raiserror 32021 "DsStCPUpdAvailableCacheSpace: The ServerId [%1!] you are attempting
to update does not exist.", @ServerId
    RETURN (@mystatus)
END

(Select @AvailableCacheSpace = isnull(AvailableCacheSpace, 0) from DsStConfigParameter
where ServerId = @ServerId)
IF (@AvailableCacheSpace - @Amount) < 0
BEGIN
    SELECT @mystatus = 1
    raiserror 32022 "DsStCPUpdAvailableCacheSpace: ServerId [%1!] has an Available Cache
Space of [%2!]. Your attempt to acquire Cache Space of [%3!] will bring the Available Cache
Space below zero. Request denied.", @ServerId, @AvailableCacheSpace, @Amount
    RETURN (@mystatus)
END

BEGIN TRANSACTION
UPDATE DsStConfigParameter
    SET AvailableCacheSpace = AvailableCacheSpace + @Amount
WHERE ServerId = @ServerId

SELECT @mystatus = @@error
IF (@mystatus != 0)
BEGIN
    ROLLBACK TRANSACTION
    SELECT @mystatus = 1
    raiserror 32023 "DsStCPUpdAvailableCacheSpace: Unable to update Available Cache
Space for ServerId [%1!].", @ServerId
    RETURN (@mystatus)
END
COMMIT TRANSACTION
RETURN(@mystatus)

END
go

```

## Procedure: DsStCPUpdStatusAndServerHandle

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStCPUpdStatusAndServerHandle
--
-- DESCRIPTION:      Updates the Status, Node and ServerHandle Fields
--                   for a ServerId
--
-- TABLES ACCESSED:  DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER          DOMAIN
--                   -----
--                   ServerId        serverid
--                   Status          status
--                   ServerHandle    path
--
-- OUTPUTS:          PARAMETER          DOMAIN
--                   -----
--                   NONE
-- ****
CREATE PROCEDURE DsStCPUpdStatusAndServerHandle
    @ServerId        serverid,
    @Status          state,
    @ServerHandle    path
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    -- Error checking to see if ServerId exists
    IF NOT EXISTS(SELECT 1 FROM DsStConfigParameter WHERE
                  ServerId = @ServerId)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32024 "DsStCPUpdStatusAndServerHandle: ServerId [%1!] does not exist.",  

@ServerId
        RETURN @mystatus
    END

    IF (@Status != "ON-LINE" and @Status != "OFF-LINE")
    BEGIN
        SELECT @mystatus = 1
        raiserror 32025 "DsStCPUpdStatusAndServerHandle: Status [%1!] is invalid. Valid status  

is ON-LINE or OFF-LINE.", @Status
    END
```

```

        RETURN @mystatus
    END

    BEGIN TRANSACTION

    IF (@Status = "OFF-LINE")
    BEGIN
        UPDATE DsStConfigParameter
        SET Status = @Status,
            Node = "",
            ServerHandle = ""
        WHERE ServerId = @ServerId
    END
    ELSE
        -- This is ON-LINE scenario
    BEGIN
        UPDATE DsStConfigParameter
        SET Status = @Status,
            Node = host_name(),
            ServerHandle = @ServerHandle
        WHERE ServerId = @ServerId
    END

    SELECT @mystatus = @@error
    IF (@mystatus !=0)
    BEGIN
        SELECT @mystatus = 1
        ROLLBACK TRANSACTION
        raiserror 32026 "DsStCPUpdStatusAndServerHandle: Unable to update Status, Node and
ServerHandle for ServerId [%1!].", @ServerId
        RETURN (@mystatus)
    END

    COMMIT TRANSACTION
    RETURN (@mystatus)

END
go

```

## Procedure: DsStDDelete

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStDDelete
--
-- DESCRIPTION:      Deletes a row from the DsStDevice Table
-- DATE CREATED: 10/3/97
-- CREATED BY:       Lisa Colombo
-- 
```

```

-- TABLES ACCESSED:  DsStConfigParameter
--
-- RETURNS:      Status (Success = 0)
--
-- INPUTS:      PARAMETER      DOMAIN
--               DeviceName      devicename
--
-- OUTPUTS:     PARAMETER      DOMAIN
--               NONE
-- ****
CREATE PROCEDURE DsStDDelete
    @DeviceName      devicename
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    -- Error checking to see if DeviceName exists
    IF NOT EXISTS(SELECT 1 FROM DsStDevice WHERE
        DeviceName = @DeviceName)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32040 "DsStDDelete: Unable to delete DeviceName [%1!] as the device does not
exist.", @DeviceName
        RETURN @mystatus
    END

    -- Error checking to see if Device is ON-LINE
    IF EXISTS(SELECT 1 FROM DsStDevice WHERE
        DeviceName = @DeviceName AND Status = 1)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32041 "DsStDDelete: Unable to delete DeviceName [%1!] as its current Status is
ON-LINE.", @DeviceName
        RETURN @mystatus
    END

    BEGIN TRANSACTION

    DELETE DsStDevice
    WHERE DeviceName = @DeviceName

    SELECT @mystatus = @@error
    IF (@mystatus !=0)
    BEGIN
        SELECT @mystatus = 1
        ROLLBACK TRANSACTION
        raiserror 32042 "DsStDDelete: Unable to delete information about DeviceName [%1!]", @DeviceName
    END

```

```

        RETURN (@mystatus)
END

COMMIT TRANSACTION
RETURN (@mystatus)

END
go

```

## Procedure: DsStDeallocate

### Code

```

-- *****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStDeallocate
--
-- DESCRIPTION: To deallocate a device for a given ScheduleId in the
--               DsStSchedule table
--               Calls the following stored procedures
--               DsStSSelectDeviceName to select DeviceName for a given
--               ScheduleId
--               DsStUpdate to update the
--                   OperationStatus as "free",
--                   CurrentStatus as "free"
--                   in the DsStDevice table.
--               DsStUpdate to update in DsStSchedule table,
--                   the value of the
--                   Status with "complete",
--               For API use.
--
-- TABLES ACCESSED:  DsStSchedule
--                  DsStDevice
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:   PARAMETER      DOMAIN
--           -----  -----
--           ScheduleId     id
--
-- OUTPUTS:  PARAMETER      DOMAIN
--           -----  -----
-- *****

```

```

CREATE PROCEDURE DsStDeallocate
    @ScheduleId      id

```

```

AS
BEGIN

```

```

DECLARE @status int
DECLARE @ss_status int
DECLARE @du_status int
DECLARE @su_status int
DECLARE @rows int
DECLARE @Device devicename
DECLARE @DeviceName devicename

SELECT @status = 0
SELECT @rows = 0

SELECT @ss_status = 0
SELECT @du_status = 0
SELECT @su_status = 0
SELECT @Device = NULL
SELECT @DeviceName = NULL

IF (@ScheduleId = 0)
BEGIN
    SELECT @status = 1
    RAISERROR 31111 "DsStDeallocate: A ScheduleId must be provided."
END -- BEGIN ScheduleId
ELSE
BEGIN
    EXECUTE @ss_status = DsStSSelectDeviceName @ScheduleId,
            @DeviceName output
    IF @ss_status != 0
    BEGIN
        SELECT @status = 1
        RAISERROR 31112 "DsStDeallocate: Error in the DsStSSelectDeviceName."
    END -- BEGIN ss_status
    ELSE
        EXECUTE @du_status = DsStDUpdate @DeviceName, 0,0
        IF @du_status != 0
        BEGIN
            SELECT @status = 1
            RAISERROR 31113 "DsStDeallocate: Error in the DsStDUpdate."
        END
        ELSE
            EXECUTE @su_status = DsStSUpdateStatus @ScheduleId,2
            IF @su_status != 0
            BEGIN
                SELECT @status = 1
                RAISERROR 31114 "DsStDeallocate: Error in the DsStSUpdateStatus. "
            END
        END
    END
    RETURN(@status)
END
go

```

## Procedure: DsStDInsert

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStDInsert
--
-- DESCRIPTION:         Insert a record into the DsStDevice table.
-- CREATE DATE:        June 12, 1997
-- DEVELOPER:          Lisa Colombo
--
-- TABLES ACCESSED:    DsStDevice
--
-- RETURNS:             Status (Success = 0)
--
-- INPUT PARAMETERS      DOMAIN
-- -----
-- DeviceName           devicename
-- Status               status
-- Capacity             capacity
-- PathName             path
-- FilePartitionPath   path
-- Model                model
-- CurrentOperation    status
-- OperationStatus     status
-- CurrentStatus       status
-- Node                 node
-- DriveNumber          drivernumber
-- DriveTapeId          drivetapeid
-- DriveCurrentSlot    drivecurrentslot
-- ElementNo            elementno
-- ServerId             serverid
-- StackerId            devicename
-- Description          description
--
-- OUTPUT PARAMETERS     DOMAIN
-- -----
-- NONE
-- ****
CREATE PROCEDURE DsStDInsert
    @DeviceName           devicename,
    @Status               status,
    @Capacity              capacity,
    @PathName             path,
    @FilePartitionPath   path,
    @Model                model,
    @CurrentOperation    status,
    @OperationStatus     status,
    @CurrentStatus       status,
    @Node                 node,
```

```

@DriveNumber           drivernumber,
@DriveTapeId          drivetapeid,
@DriveCurrentSlot     drivecurrentslot,
@ElementNo            elementno,
@ServerId             serverid,
@StackerId            devicename,
@Description          description

AS
BEGIN
    DECLARE @mystatus int

    SELECT @mystatus = 0

-- Error Checking to see if DeviceName is blank or null
    IF (@DeviceName is null or @DeviceName = "")
        BEGIN
            SELECT @mystatus = 1
            raiserror 32043 "DsStDInsert: DeviceName may not be blank or null."
        END
-- Error Checking to see if DeviceName already exists
    IF EXISTS (SELECT 1 FROM DsStDevice WHERE DeviceName = @DeviceName)
        BEGIN
            SELECT @mystatus = 1
            raiserror 32044 "DsStDevice: DeviceName [%1!] already exists.",

@DeviceName

            RETURN(@mystatus)
        END
    BEGIN TRANSACTION
    INSERT DsStDevice (
        DeviceName,
        Status,
        Capacity,
        PathName,
        FilePartitionPath,
        Model,
        CurrentOperation,
        OperationStatus,
        CurrentStatus,
        Node,
        DriveNumber,
        DriveTapeId,
        DriveCurrentSlot,
        ElementNo,
        ServerId,
        StackerId,
        Description
    )

```

```

VALUES (
    @DeviceName,
    @Status,
    @Capacity,
    @PathName,
    @FilePartitionPath,
    @Model,
    @CurrentOperation,
    @OperationStatus,
    @CurrentStatus,
    @Node,
    @DriveNumber,
    @DriveTapeId,
    @DriveCurrentSlot,
    @ElementNo,
    @ServerId,
    @StackerId,
    @Description
)

SELECT @mystatus = @@error

IF (@mystatus != 0)
BEGIN
    SELECT @mystatus = 1
    ROLLBACK TRANSACTION
    raiserror 32045 "DsStDInsert: Unable to insert DeviceName [%1!]."
    END
ELSE
BEGIN
    COMMIT TRANSACTION
    RETURN(@mystatus)
END
END
go

```

## Procedure: DsStDSelect

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStDSelect
--
-- DESCRIPTION:          Gets the complete row of information for a given
--                      ServerId from DsStDevice table.
--                      For API use.
--
-- TABLES ACCESSED:     DsStDevice
--
-- RETURNS:              Status (Success = 0)

```

```

-- INPUTS:  PARAMETER      DOMAIN
----- -----
--      ServerId      serverid

-- OUTPUTS: PARAMETER      DOMAIN
----- -----
--      ServerId      serverid,
--      DeviceName    devicename,
--      Status        status,
--      Capacity      capacity,
--      PathName      path,
--      FilePartitionPath  path,
--      Model         model,
--      CurrentOperation  status,
--      OperationStatus  status,
--      CurrentStatus   status,
--      Node          node,
--      DriveNumber    drivernumber,
--      DriveTapeId    drivetapeid,
--      DriveCurrentSlot  drivecurrentslot,
--      ElementNo      elementno,
--      StackerId     devicename,
--      Description    description
-- ****
CREATE PROCEDURE DsStDSelect
      @ServerId      devicename
AS
BEGIN
      DECLARE @status      int
      SELECT @status = 0
      IF (@ServerId = NULL OR @ServerId = "")
      BEGIN
            SELECT @status = 1
            RAISERROR 31046 "DsStDSelect: ServerId must be provided "
      END
      ELSE
            IF NOT EXISTS(SELECT 1 FROM DsStDevice WHERE
                  ServerId = @ServerId)
            BEGIN
                  SELECT @status = 1
                  RAISERROR 31047 "DsStDSelect: ServerId [% 1!] does not exist.",
                  @ServerId
            END
            ELSE
            BEGIN
                  SELECT "ServerId",      ServerId,
                        "DeviceName",    DeviceName,

```

```

    "Status",      Status,
    "Capacity",    Capacity,
    "PathName",    PathName,
    "FilePartitionPath", FilePartitionPath,
    "Model",       Model,
    "CurrentOperation", CurrentOperation,
    "OperationStatus", OperationStatus,
    "CurrentStatus",  CurrentStatus,
    "Node",        Node,
    "DriveNumber", DriveNumber,
    "DriveTapeId", DriveTapeId,
    "DriveCurrentSlot", DriveCurrentSlot,
    "ElementNo",   ElementNo,
    "StackerId",   StackerId,
    "Description", Description
FROM DsStDevice
WHERE ServerId = @ServerId

SELECT @status = @@error
IF (@status !=0)
BEGIN
    SELECT @status = 1
    RAISERROR 31048 "DsStDSelect: Unable to select for ServerId [%1!]", 
                    @ServerId
END
END

RETURN(@status)
END
go

```

## Procedure: DsStDSelectAll

### Code

```

-- *****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStDSelectAll
--
-- DESCRIPTION:        Gets the complete row of information for all
--                     devices from DsStDevice table.
--                     For API use.
--
-- TABLES ACCESSED:   DsStDevice
--
-- RETURNS:           Status (Success = 0)
--
-- INPUTS:            PARAMETER      DOMAIN
-- -----  -----  -----
--             NONE

```

```

-- OUTPUTS: PARAMETER          DOMAIN
----- -----
--     ServerId      serverid,
--     DeviceName    devicename,
--     Status        status,
--     Capacity       capacity,
--     PathName       path,
--     FilePartitionPath   path,
--     Model         model,
--     CurrentOperation  status,
--     OperationStatus  status,
--     CurrentStatus    status,
--     Node           node,
--     DriveNumber    drivenumbers,
--     DriveTapeId    drivetapeid,
--     DriveCurrentSlot  drivecurrentslot,
--     ElementNo      elementno,
--     StackerId      devicename,
--     Description     description
-- ****
CREATE PROCEDURE DsStDSelectAll
AS
BEGIN
    DECLARE @status      int

    SELECT @status = 0
    BEGIN
        SELECT "ServerId",      ServerId,
               "DeviceName",    DeviceName,
               "Status",        Status,
               "Capacity",      Capacity,
               "PathName",      PathName,
               "FilePartitionPath", FilePartitionPath,
               "Model",         Model,
               "CurrentOperation", CurrentOperation,
               "OperationStatus", OperationStatus,
               "CurrentStatus",  CurrentStatus,
               "Node",          Node,
               "DriveNumber",   DriveNumber,
               "DriveTapeId",   DriveTapeId,
               "DriveCurrentSlot", DriveCurrentSlot,
               "ElementNo",     ElementNo,
               "StackerId",     StackerId,
               "Description",   Description
        FROM  DsStDevice

        SELECT @status = @@error
        IF (@status !=0)
        BEGIN
            SELECT @status = 1

```

```

        RAISERROR 31049 "DsStDSelectAll: Unable to select information "
END

END

RETURN(@status)
END
go

```

## Procedure: DsStDSelectDeviceInfo

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStDSelectDeviceInfo
--
-- DESCRIPTION: To get the details from DsStDevice table depending on the
--               (device) server type for each of the various RequestIds
--               in DsStSchedule table.
--
-- TABLES ACCESSED:   DsStSchedule
--                   DsStDevice
--                   DsStConfigParameter
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER      DOMAIN
--                   -----
--                   ServerType     servertype
--
-- OUTPUTS:          PARAMETER      DOMAIN
--                   -----
--                   RequestId     requestid
--                   ServerType    servertype
--                   DeviceName    devicename
--                   Status        status
--                   OperationStatus status
--                   CurrentStatus  status
--                   CurrentOperation status
--                   CurrentVolume  drivetapeid
--
-- ****
CREATE PROCEDURE DsStDSelectDeviceInfo
    @ServerType      servertype
AS
BEGIN
    DECLARE @status      int
    SELECT @status = 0
    IF (@ServerType = NULL OR @ServerType = "")

```

```

BEGIN
    SELECT @status = 1
    RAISERROR 31021 "DsStDSelectDeviceInfo: ServerType must be provided."
END
ELSE
BEGIN

SELECT "RequestId",      s.RequestId,
       "ServerType",    cp.ServerType,
       "DeviceName",    d.DeviceName,
       "Status",        d.Status,
       "OperationStatus", d.OperationStatus,
       "CurrentStatus", d.CurrentStatus,
       "CurrentOperation", d.CurrentOperation,
       "CurrentVolume", d.DriveTapeId
FROM   DsStSchedule s, DsStDevice d, DsStConfigParameter cp
WHERE  s.DeviceName = d.DeviceName
AND    d.ServerId = cp.ServerId
      AND cp.ServerType = @ServerType

/*
IF (@@rowcount = 0)
BEGIN
    SELECT @status = 1
    RAISERROR 31022 "DsStDSelectDeviceInfo: No rows found for DeviceName"
END
*/
END

RETURN(@status)
END
go

```

## Procedure: DsStDSelectResPoolInfo

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStDSelectResPoolInfo
--
-- DESCRIPTION:      Selects Device details for the Resource GUI
-- DATE CREATED: 9/17/97
-- CREATED BY:      Lisa Colombo
--
-- TABLES ACCESSED:  DsStDevice
--                  DsStSchedule
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER      DOMAIN

```

```

----- -----
-- OUTPUTS:  PARAMETER      DOMAIN
----- -----
--      ServerType      servertype
--      NumDevices      smallcount
--      NumOnline       smallcount
--      NumQueued       smallcount
--
-- ****
CREATE PROCEDURE DsStDSelectResPoolInfo
AS
BEGIN
    DECLARE @status      int
    DECLARE @ServerType      servertype
    DECLARE @NumDevices      smallcount
    DECLARE @NumOnline       smallcount
    DECLARE @NumQueued       smallcount

    SELECT @status = 0

    -- Error checking to see if any ServerTypes exist. ServerTypes are
    -- prepopulated. No functionality exists to create ServerTypes.

    IF NOT EXISTS(SELECT * FROM DsStConfigParameter)
    BEGIN
        SELECT @status = 1
        raiserror 31011 "DsStDSelectResPoolInfo: No ServerTypes exist."
        RETURN (@status)
    END

    DECLARE Device_Type cursor FOR
    SELECT cp.ServerType
    FROM  DsStConfigParameter cp, DsStDevice d
    WHERE  d.ServerId = cp.ServerId

    OPEN Device_Type

    FETCH Device_Type INTO @ServerType
    WHILE (@@sqlstatus = 0)
    BEGIN

        SELECT  @NumDevices = COUNT(d.DeviceName)
        FROM   DsStDevice d, DsStConfigParameter cp
        WHERE  d.ServerId = cp.ServerId
        AND    cp.ServerType = @ServerType

        SELECT  @NumOnline = COUNT(d.Status)
        FROM   DsStDevice d, DsStConfigParameter cp
        WHERE  d.ServerId = cp.ServerId
        AND    cp.ServerType = @ServerType
    END

```

```

AND    d.Status = 0

SELECT @NumQueued = COUNT(s.Status)
FROM   DsStSchedule s, DsStDevice d, DsStConfigParameter cp
WHERE  s.DeviceName = d.DeviceName
AND    d.ServerId = cp.ServerId
AND    cp.ServerType = @ServerType
AND    s.Status = 0

SELECT "ServerType", @ServerType,
      "NumDevices", @NumDevices,
      "NumOnline",  @NumOnline,
      "NumQueued", @NumQueued

      FETCH Device_Type INTO @ServerType
END

CLOSE Device_Type
DEALLOCATE CURSOR Device_Type
RETURN(@status)
END
go

```

## Procedure: DsStDUpdate

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStDUpdate
-- DESCRIPTION:       Updates the OperationStatus with "allocated" and
--                   the CurrentStatus with "inprogress".
--                   Called from DsStScheduler stored procedure.
--
-- TABLES ACCESSED:  DsStDevice
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER      DOMAIN
-- -----  -----
--     DeviceName      devicename
--     OperationStatus status
--     CurrentStatus   status
--
-- OUTPUTS:          PARAMETER      DOMAIN
-- -----  -----
--     NONE
-- ****
CREATE PROCEDURE DsStDUpdate
    @DeviceName      devicename,
    @OperationStatus status ,

```

```

    @CurrentStatus  status
AS
BEGIN

DECLARE @status int
DECLARE @rows int

SELECT @status = 0
SELECT @rows = 0

IF @DeviceName = null
BEGIN
    SELECT @status = 1
    RAISERROR 31061 "DsStDUpdate: A DeviceName must be provided."
END
ELSE
IF @OperationStatus > 2
BEGIN
    SELECT @status = 1
    RAISERROR 31062 "DsStDUpdate: Invalid OperationStatus."
END
ELSE
IF @CurrentStatus > 2
BEGIN
    SELECT @status = 1
    RAISERROR 31063 "DsStDUpdate: Invalid CurrentStatus."
END
ELSE
BEGIN

BEGIN TRANSACTION

UPDATE DsStDevice
SET OperationStatus = @OperationStatus, CurrentStatus=@CurrentStatus
WHERE DeviceName = @DeviceName

SELECT @rows = @@rowcount
SELECT @status = @@error

IF (@status != 0)
BEGIN
    ROLLBACK TRANSACTION
    RAISERROR 31064 "DsStDUpdate: Update Failed for [%1!] in DsStDevice table.", 
                    @DeviceName
    RETURN(@status)
END

IF ( (@rows = 0) AND (@status = 0) )
BEGIN
    SELECT @status = 1
    ROLLBACK TRANSACTION
    RAISERROR 31065 "DsStDUpdate: Device Name [%1!] not in DsStDevice table."
END

```

```

@DeviceName
    RETURN(@status)
END

COMMIT TRANSACTION
END
RETURN(@status)
END
go

```

## Procedure: DsStDUpdateDevice

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStDUpdateDevice
--
-- DESCRIPTION:         Update a record in the DsStDevice table.
-- CREATE DATE:        9/22/97
-- DEVELOPER:          Lisa Colombo
--
-- TABLES ACCESSED:    DsStDevice
--
-- RETURNS:             Status (Success = 0)
--
-- INPUT PARAMETERS      DOMAIN
----- -----
--   DeviceName           devicename
--   Status                status
--   Capacity              capacity
--   PathName              path
--   FilePartitionPath     path
--   Model                 model
--   CurrentOperation      status
--   OperationStatus       status
--   CurrentStatus         status
--   Node                  node
--   DriveNumber           drivernumber
--   DriveTapeId           drivetapeid
--   DriveCurrentSlot      drivecurrentslot
--   ElementNo             elementno
--   ServerId              serverid
--   StackerId             devicename
--   Description            description
--
-- OUTPUT PARAMETERS      DOMAIN
----- -----
--   NONE
-- ****

```

```

CREATE PROCEDURE DsStDUpdateDevice
    @DeviceName          devicename,
    @Status              status,
    @Capacity             capacity,
    @PathName             path,
    @FilePartitionPath     path,
    @Model                model,
    @DriveNumber           drivernumber,
    @DriveTapeId            drivetapeid,
    @DriveCurrentSlot       drivecurrentslot,
    @ElementNo             elementno,
    @ServerId              serverid,
    @StackerId              devicename,
    @Description            description

AS
BEGIN
    DECLARE @mystatus int

    SELECT @mystatus = 0

    -- Error Checking to see if DeviceName exists
    IF NOT EXISTS (SELECT 1 FROM DsStDevice WHERE DeviceName =
    @DeviceName)
        BEGIN
            SELECT @mystatus = 1
            raiserror 32057 "DsStDUpdateDevice: DeviceName [%1!] does not exist.",

    @DeviceName
            RETURN(@mystatus)
        END

    BEGIN TRANSACTION
    UPDATE DsStDevice
    SET   DeviceName      = @DeviceName,
          Status         = @Status,
          Capacity        = @Capacity,
          PathName        = @PathName,
          FilePartitionPath = @FilePartitionPath,
          Model          = @Model,
          DriveNumber     = @DriveNumber,
          DriveTapeId      = @DriveTapeId,
          DriveCurrentSlot = @DriveCurrentSlot,
          ElementNo       = @ElementNo,
          ServerId        = @ServerId,
          StackerId       = @StackerId,
          Description      = @Description
    WHERE DeviceName = @DeviceName

    SELECT @mystatus = @@error

    IF (@mystatus != 0)
        BEGIN
            SELECT @mystatus = 1
        END

```

```

        ROLLBACK TRANSACTION
        raiserror 32058 "DsStDUpdateDevice: Unable to update information about
DeviceName [%1!].", @DeviceName
        END
    ELSE
        BEGIN
        COMMIT TRANSACTION
        RETURN(@mystatus)
    END
END
go

```

## Procedure: DsStDUpdateStatus

### Code

```

-- *****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStDUpdateStatus
--
-- DESCRIPTION:          Updates in the DsStDevice table the Status with
--                      the given status Online (0) / Offline (1).
--
-- TABLES ACCESSED:    DsStDevice
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:   PARAMETER      DOMAIN
--           -----  -----
--           DeviceName     devicename
--           Status        status
--
-- OUTPUTS:  PARAMETER      DOMAIN
--           -----  -----
--           NONE
-- *****
CREATE PROCEDURE DsStDUpdateStatus
    @DeviceName    devicename,
    @Status        status
AS
BEGIN

    DECLARE @status int
    DECLARE @rows int

    SELECT @status = 0
    SELECT @rows = 0

    IF (@DeviceName = NULL OR @DeviceName = "")
    BEGIN
        SELECT @status = 1
    END

```

```

        RAISERROR 31071 "DsStDUpdateStatus: A DeviceName must be provided."
END
ELSE
IF @Status > 1
BEGIN
    SELECT @status = 1
    RAISERROR 31072 "DsStDUpdateStatus: Invalid Status."
END
ELSE
BEGIN -- BEGIN UPDATE

BEGIN TRANSACTION

IF @Status = 0
BEGIN
UPDATE DsStDevice
SET Status = @Status
WHERE DeviceName = @DeviceName
END
ELSE
BEGIN
UPDATE DsStDevice
SET Status = @Status
WHERE DeviceName = @DeviceName
AND OperationStatus != 1
AND CurrentStatus != 1
END -- For ELSE BEGIN

SELECT @rows = @@rowcount, @status = @@error

IF (@status != 0)
BEGIN
    SELECT @status = 1
    ROLLBACK TRANSACTION
    RAISERROR 31073 "DsStDUpdateStatus: Update Failed for [%1!] in DsStDevice
table.", @DeviceName
    RETURN(@status)
END

IF (@rows = 0)
BEGIN
    SELECT @status = 1
    ROLLBACK TRANSACTION
    RAISERROR 31074 "DsStDUpdateStatus: Device Name [%1!] not in DsStDevice
table.", @DeviceName
    RETURN(@status)
END

COMMIT TRANSACTION
END -- BEGIN UPDATE

```

```

    RETURN(@status)
END
go

```

## Procedure: DsStELAlarms

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStELAlarms
--
-- DESCRIPTION:      Returns no. of Alarms since last polling date.
--                   GUI maintaining LastPollingDate.
-- CREATE DATE:       July 8, 1997
-- DEVELOPER:         Lisa Colombo
--
-- TABLES ACCESSED:   DsStEventLog
--
-- RETURNS:           Status (Success = 0)
--
-- INPUTS:            PARAMETER          DOMAIN
--                   -----  -----
--                   EventType        eventtype
--                   LastPollingDate   datetime
--
-- OUTPUTS:           PARAMETER          DOMAIN
--                   -----  -----
--                   NumberOfAlarms   int
--                   LastPollingDate   datetime
--
-- ****
CREATE PROCEDURE DsStELAlarms
    @EventType          eventtype,
    @LastPollingDate    datetime = "JAN 1, 1990"
AS
BEGIN
    DECLARE @mystatus      int
    SELECT @mystatus = 0
    -- Error checking to see if EventType is NULL
    IF (@EventType = NULL OR @EventType = "")
    BEGIN
        SELECT @mystatus = 1
        raiserror 33000 "DsStELAlarms: EventType [%1!] does not exist.", @EventType
        RETURN @mystatus
    END
    -- If LastPollingDate is blank or null default to JAN 1, 1990

```

```

IF (@LastPollingDate IS NULL or @LastPollingDate = "")
BEGIN
    SELECT @LastPollingDate = "JAN 1, 1990"
END

SELECT "NumberOfAlarms", COUNT(*),
       "LastPollingDate", CONVERT (char(20), getdate(), 9)
  FROM DsStEventLog
 WHERE EventDate > @LastPollingDate
   AND EventType = @EventType

SELECT @mystatus = @@error
IF (@mystatus != 0)
BEGIN
    SELECT @mystatus = 1
    raiserror 33001 "DsStELAlarms: Unable to select information about Event Type [%1!], Last Polling Date [%2!].", @EventType, @LastPollingDate
    RETURN (@mystatus)
END
RETURN (@mystatus)

END
go

```

## Procedure: DsStELInsert

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStELInsert
--
-- DESCRIPTION:          Inserts a record into the DsStEventLog with
--                      generate EventLogId.
--
-- TABLES ACCESSED:     DsStEventLog
--
-- RETURNS:              Status (Success = 0)
--
-- INPUTS:   PARAMETER        DOMAIN
--           -----
--           EventNumber      eventnumber
--           EventMessage     eventmessage
--           EventLevel       eventlevel
--           EventType        eventtype
--
-- OUTPUTS:  PARAMETER        DOMAIN
--           -----
--           NONE
-- ****

```

```

CREATE PROCEDURE DsStELInsert
    @EventLogId      id,
    @EventNumber     eventnumber,
    @EventMessage    eventmessage,
    @EventLevel      eventlevel,
    @EventType       eventtype
AS
BEGIN
    DECLARE @status int
    IF (@EventLogId != 0)
        BEGIN
            SELECT @status = 1
            RAISERROR 33002 "DsStELInsert: Invalid EventLogId [% 1!].",
                @EventLogId
        END
    ELSE
        IF (@EventNumber = 0)
            BEGIN
                SELECT @status = 1
                RAISERROR 33003 "DsStELInsert: EventNumber must be provided."
            END
        ELSE
            IF (@EventMessage = NULL OR @EventMessage = "")
                BEGIN
                    SELECT @status = 1
                    RAISERROR 33004 "DsStELInsert: EventMessage must be provided. "
                END
            ELSE
                IF (@EventLevel = NULL OR @EventLevel = "")
                    BEGIN
                        SELECT @status = 1
                        RAISERROR 33005 "DsStELInsert: EventLevel must be provided."
                    END
                ELSE
                    IF (@EventType = NULL OR @EventType = "")
                        BEGIN
                            SELECT @status = 1
                            RAISERROR 33006 "DsStELInsert: EventType must be provided."
                        END
                    ELSE
                        BEGIN -- BEGIN INSERT

                            BEGIN TRANSACTION

                            INSERT DsStEventLog (
                                EventLogId,
                                EventNumber,
                                EventMessage,
                                EventDate,          /* Initialised to current date */
                                EventLevel,
                                EventType
                            )

```

```

VALUES (
    @EventLogId,
    @EventNumber,
    @EventMessage,
    getdate(),
    @EventLevel,
    @EventType
)
SELECT @status = @@error
IF (@status != 0)
BEGIN
    ROLLBACK TRANSACTION
    RAISERROR 33007 "DsStELInsert: Error in Inserting into DsStEventLog."
END
ELSE
BEGIN
    COMMIT TRANSACTION
END
END -- BEGIN INSERT

RETURN (@status)
END
go

```

## Procedure: DsStELPurge

### Code

```

-- *****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStELPurge
--
-- DESCRIPTION:          To delete all the details from the DsStEventLog table
--                      that are prior to the given PurgeTime .
--
-- TABLES ACCESSED:     DsStEventLog
--
-- RETURNS:              Status (Success = 0)
--
-- INPUTS:   PARAMETER      DOMAIN
--           -----      -----
--           PurgeTime       datetime
--
-- OUTPUTS:  PARAMETER      DOMAIN
--           -----      -----
--           NONE
-- *****
CREATE PROCEDURE DsStELPurge
    @PurgeTime      datetime

```

```

AS
BEGIN

    DECLARE @status int

    SELECT @status = 0

    IF (@PurgeTime = NULL)
    BEGIN
        RAISERROR 33041 "DsStELPurge: A Purge Time must be provided. "
        SELECT @status = 1
    END
    ELSE
        IF (datediff(second,@PurgeTime,getdate()) < 0 )
        BEGIN
            RAISERROR 33042 "DsStELPurge: Purge Time must be prior to today. "
            SELECT @status = 1
        END
        ELSE

            BEGIN TRANSACTION

            DELETE
            FROM DsStEventLog
            WHERE datediff(second, EventDate ,@PurgeTime ) >0

            SELECT @status = @@error
            IF (@status != 0)
            BEGIN
                ROLLBACK TRANSACTION
                SELECT @status = 1
                RAISERROR 33043 "DsStELPurge: Unable to delete records prior to Purge Time
[%1!].",
                @PurgeTime
                RETURN @status
            END

            COMMIT TRANSACTION
            RETURN (@status)

        END
    go

```

## Procedure: DsStELSelect

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStELSelect
-- 
```

```

-- DESCRIPTION:      To get all the details from the DsStEventLog table
--
-- TABLES ACCESSED: DsStEventLog
--
-- RETURNS:         Status (Success = 0)
--
-- INPUTS:          PARAMETER           DOMAIN
-- -----          -----
-- 
-- OUTPUTS:         PARAMETER           DOMAIN
-- -----          -----
--          EventLogId      id,
--          EventNumber     eventnumber,
--          EventMessage    eventmessage,
--          EventDate       date,
--          EventLevel      eventlevel,
--          EventType       eventtype
-- 
-- ****
CREATE PROCEDURE DsStELSelect
AS
BEGIN

    DECLARE @status int

    SELECT @status = 0

    SELECT "EventLogId", EventLogId,
           "EventNumber", EventNumber,
           "EventMessage", EventMessage,
           "EventDate", EventDate,
           "EventLevel", EventLevel,
           "EventType", EventType
    FROM DsStEventLog
    ORDER BY EventDate desc

    IF (@@error != 0)
    BEGIN
        SELECT @status = 1
        RAISERROR 33009 "DsStELSelect: Error in selecting the rows."
    END

    RETURN (@status)

END
go

```

## Procedure: DsStELSelectAny

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStELSelectAny
--
-- DESCRIPTION:          Retrieves all details from the DsStEventLog table
--                      based on any combination of search criteria.
-- DATE CREATED: 9/30/97
-- CREATED BY:           Lisa Colombo
--
-- TABLES ACCESSED:    DsStEventLog
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:   PARAMETER        DOMAIN
-- -----
--          EventNumber      eventnumber,
--          EventMessage     eventmessage,
--          EventDate        datetime,
--          EventLevel       eventlevel,
--          EventType        eventtype
--
-- OUTPUTS:  PARAMETER        DOMAIN
-- -----
--          EventLogId       id,
--          EventNumber      eventnumber,
--          EventMessage     eventmessage,
--          EventDate        datetime,
--          EventLevel       eventlevel,
--          EventType        eventtype
--
-- ****
CREATE PROCEDURE DsStELSelectAny
    @EventNumber          eventnumber,
    @EventMessage         eventmessage,
    @EventDate            datetime,
    @EventLevel           eventlevel,
    @EventType            eventtype
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0
    SELECT "EventLogId", EventLogId,
          "EventNumber", EventNumber,
```

```

"EventMessage", EventMessage,
"EventDate", EventDate,
"EventLevel", EventLevel,
"EventType", EventType
FROM DsStEventLog
WHERE EventNumber = @EventNumber
OR EventMessage LIKE @EventMessage
OR EventDate = @EventDate
OR EventLevel = @EventLevel
OR EventType = @EventType
ORDER BY EventDate desc

SELECT @mystatus = @@error

IF (@mystatus != 0)
BEGIN
    SELECT @mystatus = 1
    raiserror 33010 "DsStELSelectAny: Error in selecting the rows."
    RETURN (@mystatus)
END

RETURN (@mystatus)

END
go

```

## Procedure: DsStELSelectByTime

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStELSelectByTime
--
-- DESCRIPTION:          To get all the details from the DsStEventLog table
--                      that are in the time interval between
--                      StartTime and EndTime.
--
-- DATE MODIFIED 10/1/97; LJC - Remove error message if no error
-- records found.
--
-- TABLES ACCESSED:     DsStEventLog
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:   PARAMETER      DOMAIN
--           -----
--           StartTime       datetime,
--           EndTime        datetime
--
-- OUTPUTS:  PARAMETER      DOMAIN

```

```

-- ----- -----
-- EventLogId      id,
-- EventNumber     eventnumber,
-- EventMessage    eventmessage,
-- EventDate       date,
-- EventLevel      eventlevel,
-- EventType       eventtype

-- ****
CREATE PROCEDURE DsStELSelectByTime
    @StartTime      datetime,
    @EndTime        datetime
AS
BEGIN

    DECLARE @status int

    SELECT @status = 0

    IF (@StartTime = "" OR @StartTime = NULL)
    BEGIN
        RAISERROR 33011 "DsStELSelectByTime: StartTime must be provided. "
        SELECT @status = 1
    END
    ELSE
    IF (@EndTime = "" OR @EndTime = NULL)
    BEGIN
        RAISERROR 33012 "DsStELSelectByTime: EndTime must be provided. "
        SELECT @status = 1
    END
    ELSE

        IF ( datediff(second,@StartTime,@EndTime) < 0 )
        BEGIN
            RAISERROR 33013 "DsStELSelectByTime: EndTime must be after the StartTime."
            SELECT @status = 1
        END
        ELSE

            SELECT "EventLogId", EventLogId,
                  "EventNumber", EventNumber,
                  "EventMessage", EventMessage,
                  "EventDate",   EventDate,
                  "EventLevel",  EventLevel,
                  "EventType",  EventType

            FROM DsStEventLog
            WHERE ( datediff(second, EventDate ,@StartTime ) <0
                  AND datediff(second, EventDate ,@EndTime ) >0)
            ORDER BY EventDate desc

    RETURN (@status)

```

```
END  
go
```

## Procedure: DsStELSelectByType

### Code

```
-- ****  
-- BEGIN PROLOG  
--  
-- PROCEDURE NAME:      DsStELSelectByType  
--  
-- DESCRIPTION:          To get all the details from the DsStEventLog table  
--                      for a given EventType .  
--  
-- TABLES ACCESSED:     DsStEventLog  
--  
-- RETURNS:              Status (Success = 0)  
--  
-- INPUTS:    PARAMETER           DOMAIN  
--             -----  
--             EventType        eventtype,  
--             StartTime       datetime,  
--             EndTime         datetime  
--  
-- OUTPUTS:   PARAMETER           DOMAIN  
--             -----  
--             EventLogId      id,  
--             EventNumber     eventnumber,  
--             EventMessage    eventmessage,  
--             EventDate       date,  
--             EventLevel      eventlevel,  
--             EventType       eventtype  
--  
-- ****  
CREATE PROCEDURE DsStELSelectByType  
    @EventType      eventtype,  
    @StartTime      datetime,  
    @EndTime        datetime  
AS  
BEGIN  
  
    DECLARE @status int  
  
    SELECT @status = 0  
  
    IF (@EventType = "" OR @EventType = NULL)  
    BEGIN  
        RAISERROR 33014 "DsStELSelectByType: An Event Type must be provided."  
        SELECT @status = 1  
    END
```

```

ELSE
IF (@StartTime = "" OR @StartTime = NULL)
BEGIN
    RAISERROR 33015 "DsStELSelectByType: StartTime must be provided."
    SELECT @status = 1
END
ELSE
IF (@EndTime = "" OR @EndTime = NULL)
BEGIN
    RAISERROR 33016 "DsStELSelectByType: EndTime must be provided."
    SELECT @status = 1
END
ELSE
IF ( datediff(second,@StartTime,@EndTime) < 0 )
BEGIN
    RAISERROR 33017 "DsStELSelectByType: EndTime must be after the StartTime."
    SELECT @status = 1
END
ELSE

SELECT "EventLogId", EventLogId,
       "EventNumber", EventNumber,
       "EventMessage", EventMessage,
       "EventDate", EventDate,
       "EventLevel", EventLevel,
       "EventType", EventType
FROM DsStEventLog
WHERE EventType = @EventType
AND ( datediff(second, EventDate ,@StartTime ) < 0
      AND datediff(second, EventDate ,@EndTime ) > 0 )
ORDER BY EventDate desc

IF (@@rowcount = 0)
BEGIN
    RAISERROR 33018 "DsStELSelectByType: No Records with the given EventType &
time interval."
    SELECT @status = 1
END

RETURN (@status)

END
go

```

## Procedure: DsStFLSelect

### Code

```

-- ****
-- BEGIN PROLOG
-- 
-- PROCEDURE NAME:    DsStFLSelect

```

```

-- DESCRIPTION:      Select from the DsStFileLocation table for a
--                   given FileName
--
-- TABLES ACCESSED:  DsStFileLocation
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER          DOMAIN
-- -----  -----
--     FileName        file
--     CahceId         cacheid,
--     DirectoryPath   path
--
-- OUTPUTS:          PARAMETER          DOMAIN
-- -----  -----
-- ****
CREATE PROCEDURE DsStFLSelect
    @FileName  file ,
    @CacheId   cacheid,
    @DirectoryPath path
AS
BEGIN
    DECLARE @status int

    IF @FileName = NULL
    BEGIN
        RAISERROR 30014 "DsStFLSelect: FileName must be provided"
        SELECT @status = 1
    END
    ELSE
    BEGIN

        SELECT
            "FileName",          FileName,
            "CacheId",          CacheId,
            "FileSize",          FileSize,
            "FileWeight",        FileWeight,
            "TimeOfCachePlacement", TimeOfCachePlacement,
            "TimeOfLastUse",     TimeOfLastUse,
            "CacheHitCount",    CacheHitCount,
            "ActiveLinkCount",  ActiveLinkCount,
            "AlwaysInCache",    AlwaysInCache,
            "OriginalFileName", OriginalFileName,
            "ExpirationFlag",   ExpirationFlag
        FROM  DsStCacheFile
        WHERE  FileName = @FileName

        SELECT @status = @@error
    END

```

```

IF (@status != 0)
BEGIN
RAISERROR 30018 "DsStFLSelect: Unable to select information about FileName [%1!]", 
@FileName
RETURN (@status)
END

RETURN (@status)
END
RETURN (@status)
END
go

```

## Procedure: DsStNextIdGet

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStNextIdGet
--
-- DESCRIPTION: To get the next available number for a table
--               and increment the number for the next record.
--
-- TABLES ACCESSED:    DsStNextId
--
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER      DOMAIN
--                   -----      -----
--                   TableName      tablename
--
-- OUTPUTS:          PARAMETER      DOMAIN
--                   -----      -----
--                   NextId        id
--
-- ****

```

```

CREATE PROCEDURE DsStNextIdGet
    @TableName      tablename,
    @IDnumber int = 0 output

```

```

AS
BEGIN
    DECLARE @status int
    DECLARE @new_next_id int

```

```

    SELECT @status = 0

```

```

    IF @TableName = null
    BEGIN

```

```

SELECT @status = 1
      RAISERROR 30001 "DsStNextIdGet: A Table Name must be provided."
END
ELSE
BEGIN

-- Retrieve the next available id for the table

SELECT @IDnumber = NextId
FROM DsStNextId
WHERE TableName = @TableName

IF @@rowcount = 0
BEGIN
  SELECT @status = 2
  RAISERROR 30002 "DsStNextIdGet: Table named [%1!] is not entered in
DsStNextId table.",

@TableName
END
-- Increment and update the id number for the specified table

SELECT @new_next_id = @IDnumber +1

UPDATE DsStNextId
SET NextId = @new_next_id
WHERE TableName = @TableName

IF (@@error != 0)
BEGIN
  SELECT @status = 1
  RAISERROR 30003 "DsStNextIdGet: Next id value of [%1!] cannot be inserted into
DsStNextId table.",

@IDnumber
END
END -- for BEGIN ELSE
RETURN(@status)
END
go

```

## **Procedure: DsStNextIdInsert**

### **Code**

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:  DsStNextIdInsert
--
-- DESCRIPTION:      Insert default data into database
-- 
```

```

-- TABLES ACCESSED:  DsStNextId
--
-- RETURNS:          status (success = 0)
--
-- INPUTS  PARAMETER    DOMAIN
-- -----  -----
--      TableName   tablename
--      NextId      id
--
-- OUTPUTS PARAMETER    DOMAIN
-- -----  -----
--      NONE
-- ****
CREATE PROCEDURE DsStNextIdInsert
    @TableName tablename,
    @NextId   id
AS
BEGIN
    DECLARE @status int
    SELECT @status = 0
    BEGIN TRANSACTION
    INSERT DsStNextId (TableName, NextId )
        values (@TableName, @NextId)
    SELECT @status = @@error
    IF (@status != 0)
        BEGIN
            ROLLBACK TRANSACTION
            SELECT @status = 1
            RETURN @status
        END
    COMMIT TRANSACTION
END
go

```

## Procedure: DsStPLSelectByExpTime

### Code

```

-- ****
-- BEGIN PROLOG
-- PROCEDURE NAME:    DsStPLSelectByExpTime

```

```

-- DESCRIPTION:      Retrieves all details of files for the PullMonitor
--                  that are expired.
--                  Calls DsStPLUpdateExpirationFlag stored procedure
--                  to get the details including latest expired files.
--
-- TABLES ACCESSED: pull_list
--
-- RETURNS:          Status (Success = 0)
--                  Returns all rows in pull_list table that
--                  are in the PullExpirationTime for PullMonitor,
--                  with the ExpirationFlag as "EXPIRED"
--
-- INPUTS:           PARAMETER          DOMAIN
--                   -----      -----
--                   ServerId       serverid
--
-- OUTPUTS:          PARAMETER          DOMAIN
--                   -----      -----
--                   uniq_file     varchar(255),
--                   time_stamp    datetime,
--                   file_size     int,
--                   acc_count     int,
--                   ExpirationFlag expirationflag
--
-- ****
CREATE PROCEDURE DsStPLSelectByExpTime
    @ServerId      serverid
AS
BEGIN
    DECLARE @status int
    DECLARE @pu_status int /* status for pull_list Update Procedure */
    DECLARE @ExpTime int

    SELECT @status = 0
    SELECT @pu_status = 0
    IF (@ServerId = NULL OR @ServerId = "")
        BEGIN
            SELECT @status = 1
            RAISERROR 31021 "DsStPLselectByExpTime: ServerId must be provided."
        END
    ELSE
        BEGIN
            EXECUTE @pu_status = DsStPLUpdateExpirationFlag @ServerId
            IF @pu_status != 0
                BEGIN
                    SELECT @status = 1
                    RAISERROR 31037 "DsStPLUpdateExpirationFlag: Error in the

```

```

DsStPLUpdateExpirationFlag. "
END
ELSE

    SELECT "uniq_file", uniq_file,
           "time_stamp", time_stamp,
           "file_size", file_size,
           "acc_count", acc_count,
           "ExpirationFlag", ExpirationFlag
    FROM pull_list
   WHERE ExpirationFlag = "EXPIRED"

    SELECT @status = @@error
  IF (@status != 0)
  BEGIN
    SELECT @status = 1
    RAISERROR 34013 "DsStPLSelectByExpTime: Unable to select pull_list records."
    RETURN(@status)
  END

  RETURN (@status)
END
go

```

## Procedure: DsStPLUpdateExpirationFlag

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStPLUpdateExpirationFlag
--
-- DESCRIPTION: To update the expiration flag in pull_list table
--               for the PullMonitor .
--               Called from DsStPLSelectByExpTime
--
-- TABLES ACCESSED:    pull_list
--                   DsStConfigParameter
--
-- RETURNS:      Status (Success = 0)
--               Updates all rows in pull_list table that
--               are in the ExpirationThreshold for PullMonitor,
--               and sets the ExpirationFlag to "EXPIRED"
--
-- INPUTS:      PARAMETER      DOMAIN
--               -----
--               ServerId       serverid
--
-- OUTPUTS:     PARAMETER      DOMAIN

```

```

----- -----
-- ****
CREATE PROCEDURE DsStPLUpdateExpirationFlag
    @ServerId      serverid

AS
BEGIN

    DECLARE @status int
    DECLARE @rows int
    DECLARE @ExpTime int

    SELECT @status = 0
    SELECT @rows = 0

    SELECT @ExpTime = PullExpirationTime from DsStConfigParameter
    WHERE ServerId = @ServerId
    IF (@@rowcount = 0)
        BEGIN
            SELECT @status = 1
            RAISERROR 34014 "DsStPLUpdateExpirationFlag: No Records for Pull Monitor
Server[%1!].", @ServerId
            END
        ELSE
            BEGIN -- BEGIN ELSE

                BEGIN TRANSACTION
                    UPDATE pull_list
                    SET ExpirationFlag = "EXPIRED"
                    WHERE datediff(hour, time_stamp ,getdate()) > @ExpTime

                    IF (@@error != 0)
                        BEGIN
                            SELECT @status = 1
                            ROLLBACK TRANSACTION
                            RAISERROR 34015 "DsStPLUpdateExpirationFlag: Error in updating. "
                        END

                COMMIT TRANSACTION
            END -- END ELSE
            RETURN (@status)
        END
    END
go

```

## Procedure: DsStScheduler

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStScheduler
--
-- DESCRIPTION: To allocate a device for a request in the schedule table
--               depending on Priority, StartDate and Status as "queued".
--               Also calls the following stored procedures:
--               DsStDSelect      to select the device available from
--                               DsStDevice table with
--                               Status as "online"
--                               OperationStatus as "free" and
--                               CurrentStatus as "free" for the
--                               particular DeviceType.
--               If found one, calls the following stored procedures:
--               DsStDUpdate      to update the
--                               OperationStatus as "allocated",
--                               CurrentStatus as "inprogress"
--                               in the DsStDevice table.
--               DsStSUpdate      to update in DsStSchedule table,
--                               the values of the
--                               DeviceName with DeviceName in the
--                               DsStDevice table from the output
--                               of DsStDSelect stored procedure,
--                               Status as "inprogress", StartDate,
--                               EstStartDate as the time the
--                               DeviceName is returned from
--                               DsStDSelect, EstmatedEndDate as
--                               the time the DeviceName is
--                               from DsStDSelect + duration,
--                               ServerHandle as the CDS path for
--                               the Node whose DeviceName is returned
--                               from DsStDSelect
--
-- TABLES ACCESSED:  DsStSchedule
--                   DsStDevice
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER          DOMAIN
--                   -----
--                   ServerType        servertype
--                   ServerHandle      path
--                   -----
-- OUTPUTS:          PARAMETER          DOMAIN
--                   -----
--
```

```

-- ****
CREATE PROCEDURE DsStScheduler
    @ServerType      servertype,
    @ServerHandle    path

AS
BEGIN
    DECLARE @status int
    DECLARE @ds_status int /* status for Device Select Procedure */
    DECLARE @du_status int /* status for Device Update Procedure */
    DECLARE @su_status int /* status for Schedule Update Procedure */
    DECLARE @ScheduleId id
    DECLARE @Node node
    DECLARE @DeviceName devicename

    SELECT @status = 0
    SELECT @ds_status = 0
    SELECT @du_status = 0
    SELECT @su_status = 0
    SELECT @ScheduleId = 0
    SELECT @DeviceName=NULL

    SELECT @Node = host_name()

    IF (@Node = NULL)
    BEGIN
        SELECT @status = 1
        RAISERROR 31031 "DsStScheduler: A Node must be provided."
    END
    ELSE
    IF (@ServerType = NULL)
    BEGIN
        SELECT @status = 1
        RAISERROR 31032 "DsStScheduler: A Server Type must be provided."
    END
    ELSE
    IF (@ServerHandle = NULL)
    BEGIN
        RAISERROR 31033 "DsStScheduler: A ServerHandle must be provided."
        SELECT @status = 1
    END
    ELSE
    BEGIN

        DECLARE Dev_Sch cursor FOR
        SELECT ScheduleId
        FROM DsStSchedule s, DsStDevice d, DsStConfigParameter cp
        WHERE s.Status = 0
        AND s.ServerId = cp.ServerId
        AND cp.ServerType = @ServerType
        AND s.DeviceName is NULL
    END
END

```

```

OPEN Dev_Sch
IF (@@error != 0)
BEGIN
    SELECT @status = 1
    RAISERROR 31034 "DsStScheduler: Error in Opening cursor. "
END

FETCH Dev_Sch INTO @ScheduleId
IF (@@error != 0)
BEGIN
    SELECT @status = 1
    RAISERROR 31035 "DsStScheduler: Error in Fetch cursor. "
END
WHILE (@@sqlstatus =0)
BEGIN
    SELECT @DeviceName = NULL

EXECUTE @ds_status = DsStDSelect @ServerType, @Node, @DeviceName output
IF @ds_status != 0
BEGIN
    SELECT @status = 1
    RAISERROR 31036 "DsStScheduler: Error in the DsStDSelect. "
END
ELSE
IF (@DeviceName is not NULL)
BEGIN
-- BEGIN TRANSACTION
    EXECUTE @du_status = DsStDUpdate @DeviceName, 1,2
    IF @du_status != 0
    BEGIN
        SELECT @status = 1
        RAISERROR 31037 "DsStScheduler: Error in the DsStDUpdate. "
    END
    ELSE
        EXECUTE @su_status =
            DsStSUpdate @ScheduleId, @DeviceName, @ServerHandle
        IF @su_status != 0
        BEGIN
            SELECT @status = 1
            RAISERROR 31038 "DsStScheduler: Error in the DsStSUpdate. "
        END
    /*
    IF (@status != 0)
    BEGIN
        ROLLBACK TRANSACTION
        RETURN (@status)
    END
    ELSE
    BEGIN
        COMMIT TRANSACTION
    */
END

```

```

    END

*/
    END
    FETCH Dev_Sch INTO @ScheduleId
    END

    CLOSE Dev_Sch
    DEALLOCATE CURSOR Dev_Sch
    END
    RETURN(@status)
END
go

```

## Procedure: DsStSInsert

### Code

```

-- *****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStSInsert
--
-- DESCRIPTION:         Insert a record into the DsStSchedule table.
--
-- TABLES ACCESSED:    DsStSchedule
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:   PARAMETER      DOMAIN
--           -----      -----
--           ScheduleId     id,
--           RequestId      requestid,
--           Priority        priority,
--           ServerId        serverid,
--           EstimatedDuration duration,
--
-- OUTPUTS:  PARAMETER      DOMAIN
--           -----      -----
--           ScheduleId     id
--
-- *****

```

```

CREATE PROCEDURE DsStSInsert
    @ScheduleId      id,
    @RequestId       requestid,
    @Priority        priority,
    @ServerId        serverid,
    @EstimatedDuration duration
AS
BEGIN
    DECLARE @status int

```

```

DECLARE @SId id

IF @ScheduleId != 0
BEGIN
    RAISERROR 31006 "DsStSInsert: Invalid ScheduleId [%1!].",
                    @ScheduleId
    SELECT @status = 1
END
ELSE
IF @RequestId = NULL
BEGIN
    RAISERROR 31001 "DsStSInsert: RequestId must be provided"
    SELECT @status = 1
END
ELSE
IF @Priority >2
BEGIN
    RAISERROR 31002 "DsStSInsert: Invalid Priority [%1!].", @Priority
    SELECT @status = 1
END
ELSE
IF @ServerId = NULL
BEGIN
    RAISERROR 31003 "DsStSInsert: ServerId must be provided"
    SELECT @status = 1
END
ELSE
IF @EstimatedDuration = 0
BEGIN
    RAISERROR 31004 "DsStSInsert: Duration must be provided "
    SELECT @status = 1
END
ELSE

BEGIN TRANSACTION

INSERT DsStSchedule (
    ScheduleId,
    Status, /* Initialized to 0- queued */
    RequestId,
    EstimatedDuration,
    StartDate, /* Initialized to current date */
    EndDate, /* Initialized to current date */
    EstimatedStartDate, /* Initialized to current date */
    EstimatedEndDate, /* Initialized to current date */
    Priority,
    ServerId
)
VALUES (
    @ScheduleId,
    0,
    @RequestId,

```

```

        @EstimatedDuration,
        getdate(),
        getdate(),
        getdate(),
        getdate(),
        @Priority,
        @ServerId
    )

SELECT @status = @@error
IF (@status != 0)
BEGIN
RAISERROR 31005 "DsStSInsert: Error inserting into DsStSchedule"
    ROLLBACK TRANSACTION
    RETURN (@status)
END
ELSE
BEGIN
SELECT @SId = ScheduleId
FROM DsStSchedule s, DsStDevice d, DsStConfigParameter cp
WHERE s.ServerId      = cp.ServerId
AND cp.ServerId      = @ServerId
    AND s.Priority      = @Priority
AND s.RequestId      = @RequestId
AND s.EstimatedDuration = @EstimatedDuration

SELECT "ScheduleId", @SId
SELECT @SId = NextId-1 FROM DsStNextId
    WHERE TableName = "DsStSchedule"
SELECT "ScheduleId", @SId
    COMMIT TRANSACTION
    RETURN (@status)
END
RETURN (@status)
END
go

```

## Procedure: DsStSKDelete

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStSKDelete
--
-- DESCRIPTION:    Deletes a row from the DsStStacker Table
-- DATE CREATED: 10/3/97
-- CREATED BY:      Lisa Colombo
--
-- TABLES ACCESSED:  DsStStacker
-- 
```

```

-- RETURNS:      Status (Success = 0)
--
-- INPUTS:  PARAMETER      DOMAIN
--          -----
--          StackerId      devicename
--
-- OUTPUTS:  PARAMETER      DOMAIN
--          -----
--          NONE
-- ****
CREATE PROCEDURE DsStSKDelete
    @StackerId      devicename
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    -- Error checking to see if StackerId exists
    IF NOT EXISTS(SELECT 1 FROM DsStStacker WHERE
        StackerId = @StackerId)
        BEGIN
            SELECT @mystatus = 1
            raiserror 32300 "DsStSKDelete: Unable to Delete StackerId [%1!] as the Stacker does not
exist.", @StackerId
            RETURN @mystatus
        END

    -- Error checking to see if Stacker has any Slots
    IF EXISTS (SELECT 1 FROM DsStSlot WHERE
        StackerId = @StackerId)
        BEGIN
            SELECT @mystatus = 1
            raiserror 32301 "DsStSKDelete: Unable to Delete StackerId [%1!] as Stacker has Slots
current defined to it. Delete Stacker's Slots first.", @StackerId
            RETURN @mystatus
        END

    -- Error checking to see if Device exists to Stacker
    IF EXISTS (SELECT 1 FROM DsStDevice WHERE
        StackerId = @StackerId)
        BEGIN
            SELECT @mystatus = 1
            raiserror 32302 "DsStSKDelete: Unable to Delete StackerId [%1!] as a device drive is
current assigned to it. Either delete the Device or assign a different StackerId.", @StackerId
            RETURN @mystatus
        END

    BEGIN TRANSACTION

    DELETE DsStStacker
    WHERE StackerId = @StackerId

```

```

SELECT @mystatus = @error
IF (@mystatus !=0)
BEGIN
    SELECT @mystatus = 1
    ROLLBACK TRANSACTION
    raiserror 32303 "DsStSKDelete: Unable to delete information about StackerId [%1!]",  

@StackerId
    RETURN (@mystatus)
END

COMMIT TRANSACTION
RETURN (@mystatus)

END
go

```

## Procedure: DsStSKInsert

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStSKInsert
--
-- DESCRIPTION:         Insert a record into the DsStStacker table.
-- DATE CREATED:        7/1/97
-- CREATED BY:          Lisa Colombo
--
-- TABLES ACCESSED:     DsStStacker
--
-- RETURNS:              Status (Success = 0)
--
-- INPUT PARAMETERS      DOMAIN
----- -----
--      StackerId        devicename
--      ErrorCount       smallcount
--      TotalSlots        smallcount
--      OnlineSlots       smallcount
--      TotalDrives       smallcount
--      OnlineDrives      smallcount
--      TotalRoSlots      smallcount
--      TotalRwSlots      smallcount
--      StackerPath       path
--
-- OUTPUT PARAMETER      DOMAIN
----- -----
--      NONE
-- ****
CREATE PROCEDURE DsStSKInsert
    @StackerId           devicename,

```

```

        @ErrorCount          smallcount,
        @TotalSlots           smallcount,
        @OnlineSlots          smallcount,
        @TotalDrives          smallcount,
        @OnlineDrives         smallcount,
        @TotalRoSlots         smallcount,
        @TotalRwSlots         smallcount,
        @StackerPath          path

AS
BEGIN
    DECLARE @mystatus int

    SELECT @mystatus = 0

-- Error Checking to see if StackerId is blank or null
    IF (@StackerId is null or @StackerId = "")
        BEGIN
            SELECT @mystatus = 1
            raiserror 32304 "DsStSKInsert: StackerId may not be blank or null."
            RETURN(@mystatus)
        END
-- Error Checking to see if StackerId already exists
    IF EXISTS (SELECT 1 FROM DsStStacker WHERE StackerId = @StackerId)
        BEGIN
            SELECT @mystatus = 1
            raiserror 32305 "DsStStacker: StackerId [% 1!] already exists.", @StackerId
            RETURN(@mystatus)
        END
    BEGIN TRANSACTION
    INSERT DsStStacker (
        StackerId,
        ErrorCount,
        TotalSlots,
        OnlineSlots,
        TotalDrives,
        OnlineDrives,
        TotalRoSlots,
        TotalRwSlots,
        StackerPath
    )
    VALUES (
        @StackerId,
        @ErrorCount,
        @TotalSlots,
        @OnlineSlots,
        @TotalDrives,
        @OnlineDrives,
        @TotalRoSlots,

```

```

        @TotalRwSlots,
        @StackerPath
    )

SELECT @mystatus = @@error

IF (@mystatus != 0)
BEGIN
    SELECT @mystatus = 1
    ROLLBACK TRANSACTION
    raiserror 32306 "DsStSKInsert: Unable to insert StackerId [%1!]."
    END
ELSE
BEGIN
    COMMIT TRANSACTION
    RETURN(@mystatus)
END
END
go

```

## Procedure: DsStSKSelect

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStSKSelect
--
-- DESCRIPTION:          Gets the complete row of information for a given
--                      StackerId from DsStStacker table.
--                      For API use.
--
-- TABLES ACCESSED:     DsStStacker
--
-- RETURNS:              Status (Success = 0)
--
-- INPUTS:   PARAMETER      DOMAIN
--           -----
--           StackerId       devicename
--
-- OUTPUTS:  PARAMETER      DOMAIN
--           -----
--           StackerId       devicename,
--           ErrorCount      smallcount,
--           TotalSlots      smallcount ,
--           OnlineSlots     smallcount,
--           TotalDrives     smallcount,
--           OnlineDrives    smallcount,
--           TotalRoSlots    smallcount,
--           TotalRwSlots    smallcount,
--           StackerPath     path

```

```

-- ****
CREATE PROCEDURE DsStSKSelect
    @StackerId      devicename
AS
BEGIN
    DECLARE @status          int
    SELECT @status = 0

    IF (@StackerId = NULL OR @StackerId = "")
    BEGIN
        SELECT @status = 1
        RAISERROR 32309 "DsStSKSelect: StackerId must be provided"
    END
    ELSE
    IF NOT EXISTS(SELECT 1 FROM DsStDevice WHERE
        StackerId = @StackerId)
    BEGIN
        SELECT @status = 1
        RAISERROR 32310 "DsStSKSelect: StackerId [% 1!] does not exist.",,
                        @StackerId
    END
    ELSE
    BEGIN
        SELECT "StackerId",      StackerId,
               "ErrorCount",     ErrorCount,
               "TotalSlots",     TotalSlots,
               "OnlineSlots",    OnlineSlots,
               "TotalDrives",   TotalDrives,
               "OnlineDrives",  OnlineDrives,
               "TotalRoSlots",  TotalRoSlots,
               "TotalRwSlots",  TotalRwSlots,
               "StackerPath",    StackerPath
        FROM  DsStStacker
        WHERE StackerId = @StackerId

        SELECT @status = @@error
        IF (@status !=0)
        BEGIN
            SELECT @status = 1
            RAISERROR 32311 "DsStSKSelect: Unable to select for StackerId [% 1!]",,
                            @StackerId
        END
    END
    RETURN(@status)
END
go

```

## Procedure: DsStSKSelectAll

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStSKSelectAll
--
-- DESCRIPTION:          Gets the complete row of information for all
--                      Stackers from DsStStacker table.
--                      For API use.
--
-- TABLES ACCESSED:    DsStStacker
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:   PARAMETER      DOMAIN
--           -----      -----
--           NONE
--
-- OUTPUTS:  PARAMETER      DOMAIN
--           -----      -----
--           StackerId      devicename,
--           ErrorCount     smallcount,
--           TotalSlots     smallcount ,
--           OnlineSlots    smallcount,
--           TotalDrives    smallcount,
--           OnlineDrives   smallcount,
--           TotalRoSlots   smallcount,
--           TotalRwSlots   smallcount,
--           StackerPath    path
--
-- ****
CREATE PROCEDURE DsStSKSelectAll
AS
BEGIN
    DECLARE @status      int
    SELECT @status = 0
    BEGIN
        SELECT "StackerId",      StackerId,
               "ErrorCount",    ErrorCount,
               "TotalSlots",    TotalSlots,
               "OnlineSlots",   OnlineSlots,
               "TotalDrives",   TotalDrives,
               "OnlineDrives",  OnlineDrives,
               "TotalRoSlots",  TotalRoSlots,
               "TotalRwSlots",  TotalRwSlots,
               "StackerPath",   StackerPath
    END
END
```

```

FROM  DsStStacker

SELECT @status = @@error
IF (@status !=0)
BEGIN
    SELECT @status = 1
    RAISERROR 32312 "DsStSKSelect: Unable to select information"
END
END
RETURN(@status)
END
go

```

## **Procedure: DsStSKUpdate**

### **Code**

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStSKUpdate
--
-- DESCRIPTION:          Update a record in the DsStStacker table.
-- DATE CREATED: 9/22/97
-- CREATED BY:           Lisa Colombo
--
-- TABLES ACCESSED:     DsStStacker
--
-- RETURNS:              Status (Success = 0)
--
-- INPUT PARAMETERS        DOMAIN
----- -----
--      StackerId          devicename
--      ErrorCount         smallcount
--      TotalSlots          smallcount
--      OnlineSlots         smallcount
--      TotalDrives         smallcount
--      OnlineDrives        smallcount
--      TotalRoSlots        smallcount
--      TotalRwSlots        smallcount
--      StackerPath         path
--
-- OUTPUT PARAMETER        DOMAIN
----- -----
--      NONE
-- ****

CREATE PROCEDURE DsStSKUpdate
    @StackerId          devicename,
    @ErrorCount         smallcount,
    @TotalSlots          smallcount,
    @OnlineSlots         smallcount,
    @TotalDrives         smallcount,

```

```

@OnlineDrives          smallcount,
@TotalRoSlots          smallcount,
@TotalRwSlots          smallcount,
@StackerPath           path
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

-- Error Checking to see if StackerId exists
    IF NOT EXISTS (SELECT 1 FROM DsStStacker WHERE StackerId = @StackerId)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32307 "DsStStacker: StackerId [%1!] does not exists.", @StackerId

        RETURN(@mystatus)
    END

    BEGIN TRANSACTION
    UPDATE   DsStStacker
    SET     StackerId      = @StackerId,
            ErrorCode     = @ErrorCode,
            TotalSlots    = @TotalSlots,
            OnlineSlots   = @OnlineSlots,
            TotalDrives   = @TotalDrives,
            OnlineDrives  = @OnlineDrives,
            TotalRoSlots  = @TotalRoSlots,
            TotalRwSlots  = @TotalRwSlots,
            StackerPath   = @StackerPath
    WHERE StackerId = @StackerId

    SELECT @mystatus = @@error

    IF (@mystatus != 0)
    BEGIN
        SELECT @mystatus = 1
        ROLLBACK TRANSACTION
        raiserror 32308 "DsStSKUpdate: Unable to update information about StackerId
[%1!].", @StackerId
        END
    ELSE
        BEGIN
            COMMIT TRANSACTION
            RETURN(@mystatus)
        END
    END
go

```

## Procedure: DsStSLDelete

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStSLDelete
--
-- DESCRIPTION:      Deletes a row from the DsStSlot Table
-- DATE CREATED: 10/3/97
-- CREATED BY:        Lisa Colombo
--
-- TABLES ACCESSED:  DsStSlot
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:    PARAMETER          DOMAIN
--           -----
--           StackerId       devicename
--           SlotNumber      slotnumber
--
-- OUTPUTS:   PARAMETER          DOMAIN
--           -----
--           NONE
-- ****
CREATE PROCEDURE DsStSLDelete
    @StackerId  devicename,
    @SlotNumber slotnumber
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    -- Error checking to see if Slot record exists
    IF NOT EXISTS(SELECT 1 FROM DsStSlot WHERE
                  StackerId = @StackerId AND SlotNumber = @SlotNumber)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32400 "DsStSLDelete: Unable to delete StackerId [%1!], SlotNumber [%2!] as
the record does not exist.", @StackerId, @SlotNumber
        RETURN @mystatus
    END

    -- Error checking to see if Slot is ON-LINE
    IF EXISTS(SELECT 1 FROM DsStSlot WHERE
                  StackerId = @StackerId AND SlotNumber = @SlotNumber
                  AND SlotStatus = 1)
    BEGIN
        SELECT @mystatus = 1
```

```

raiserror 32401 "DsStSLDelete: Unable to Delete StackerId [%1!] and SlotNumber [%2!]
as its current SlotStatus is IN USE.", @StackerId, @SlotNumber
    RETURN @mystatus
END

BEGIN TRANSACTION

DELETE DsStSlot
WHERE StackerId = @StackerId AND SlotNumber = @SlotNumber

SELECT @mystatus = @@error
IF (@mystatus !=0)
BEGIN
    SELECT @mystatus = 1
    ROLLBACK TRANSACTION
    raiserror 32402 "DsStSLDelete: Unable to delete information about StackerId [%1!] and
SlotNumber [%2!].", @StackerId, @SlotNumber
    RETURN (@mystatus)
END

COMMIT TRANSACTION
RETURN (@mystatus)

END
go

```

## Procedure: DsStSLInsert

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStSLInsert
--
-- DESCRIPTION:         Insert a record into the DsStSlot table.
-- DATE CREATED:       7/1/97
-- CREATED BY:          Lisa Colombo
--
-- TABLES ACCESSED:    DsStSlot
--
-- RETURNS:             Status (Success = 0)
--
-- INPUT PARAMETERS           DOMAIN
-- -----
--   Stackerid            devicename
--   SlotNumber           slotnumber
--   SlotStatus           status
--   SlotTapeId           slottapeid
--   SlotUse               status
--   SlotCurrentDrive     smallcount
--   ElementNo            elementno

```

```

--      SlotCapacity      capacity
--      SlotCurrentStatus status
--
-- OUTPUT PARAMETERS      DOMAIN
----- ----- -----
--      NONE
-- ****
CREATE PROCEDURE DsStSLInsert
    @StackerId          devicename,
    @SlotNumber          slotnumber,
    @SlotStatus          status,
    @SlotTapeId          slottapeid,
    @SlotUse              status,
    @SlotCurrentDrive   smallcount,
    @ElementNo           elementno,
    @SlotCapacity         capacity,
    @SlotCurrentStatus   status
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    -- Error Checking to see if StackerId is blank or null
    IF (@StackerId is null or @StackerId= '')
        BEGIN
            SELECT @mystatus = 1
            raiserror 32403 "DsStSLInsert: StackerId may not be blank or null."
            RETURN(@mystatus)
        END
    -- Error Checking to see if StackerId & SlotNumber already exists
    IF EXISTS (SELECT 1 FROM DsStSlot WHERE StackerId = @StackerId AND
    SlotNumber = @SlotNumber)
        BEGIN
            SELECT @mystatus = 1
            raiserror 32404 "DsStSLInsert: StackerId [%1!] & SlotNumber [%2!] already
exists.", @StackerId, @SlotNumber
            RETURN(@mystatus)
        END
    BEGIN TRANSACTION
    INSERT DsStSlot (
        StackerId,
        SlotNumber,
        SlotStatus,
        SlotTapeId,
        SlotUse,
        SlotCurrentDrive,
        ElementNo,
        SlotCapacity,

```

```

        SlotCurrentStatus
    )

VALUES (
    @StackerId,
    @SlotNumber,
    @SlotStatus,
    @SlotTapeId,
    @SlotUse,
    @SlotCurrentDrive,
    @ElementNo,
    @SlotCapacity,
    @SlotCurrentStatus
)

SELECT @mystatus = @@error

IF (@mystatus != 0)
BEGIN
    SELECT @mystatus = 1
    ROLLBACK TRANSACTION
    raiserror 32405 "DsStSLInsert: Unable to insert StackerId [%1!] & SlotNumber
    [%2!]."
    END
ELSE
BEGIN
    COMMIT TRANSACTION
    RETURN(@mystatus)
END
END
go

```

## Procedure: DsStSLSelect

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStSLSelect
--
-- DESCRIPTION:          Gets the complete row of information for a given
--                      StackerId from DsStSlot table.
--                      For API use.
--
-- TABLES ACCESSED:     DsStSlot
--
-- RETURNS:              Status (Success = 0)
--
-- INPUTS:   PARAMETER      DOMAIN
--           -----      -----
--           StackerId      devicename

```

```

-- OUTPUTS:  PARAMETER          DOMAIN
-----  -----
--      StackerId      devicename,
--      SlotNumber     slotnumber,
--      SlotStatus      status ,
--      SlotTapeId      slottapeid,
--      SlotUse         status,
--      SlotCurrentDrive smallcount,
--      ElementNo       elementno,
--      SlotCapacity     capacity,
--      SlotCurrentStatus status
-- ****
CREATE PROCEDURE DsStSLSelect
    @StackerId      devicename
AS
BEGIN
    DECLARE @status          int
    SELECT @status = 0
    IF (@StackerId = NULL OR @StackerId = "")
        BEGIN
            SELECT @status = 1
            RAISERROR 32408 "DsStSLSelect: StackerId must be provided "
        END
        ELSE
        IF NOT EXISTS(SELECT 1 FROM DsStDevice WHERE
                      StackerId = @StackerId)
            BEGIN
                SELECT @status = 1
                RAISERROR 32409 "DsStSLSelect: StackerId [% 1!] does not exist.",
                                @StackerId
            END
            ELSE
            BEGIN
                SELECT "StackerId",      StackerId,
                      "SlotNumber",     SlotNumber,
                      "SlotStatus",     SlotStatus,
                      "SlotTapeId",     SlotTapeId,
                      "SlotUse",        SlotUse,
                      "SlotCurrentDrive", SlotCurrentDrive,
                      "ElementNo",      ElementNo,
                      "SlotCapacity",   SlotCapacity,
                      "SlotCurrentStatus", SlotCurrentStatus
                FROM  DsStSlot
                WHERE StackerId = @StackerId
                SELECT @status = @@error
                IF (@status !=0)

```

```

BEGIN
    SELECT @status = 1
    RAISERROR 32410 "DsStSLSelect: Unable to select for StackerId [%1!]", 
                    @StackerId
END

END

RETURN(@status)
END
go

```

## Procedure: DsStSLUpdate

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStSLUpdate
--
-- DESCRIPTION:          Update a record in the DsStSlot table.
-- DATE CREATED:        9/22/97
-- CREATED BY:          Lisa Colombo
--
-- TABLES ACCESSED:     DsStSlot
--
-- RETURNS:              Status (Success = 0)
--
-- INPUT PARAMETERS      DOMAIN
-- -----
--   StackerId           devicename
--   SlotNumber          slotnumber
--   SlotStatus          status
--   SlotTapeId          slottapeid
--   SlotUse              status
--   SlotCurrentDrive    smallcount
--   ElementNo            elementno
--   SlotCapacity         capacity
--   SlotCurrentStatus   status
--
-- OUTPUT PARAMETERS     DOMAIN
-- -----
--   NONE
-- ****

CREATE PROCEDURE DsStSLUpdate
    @StackerId           devicename,
    @SlotNumber          slotnumber,
    @SlotStatus          status,
    @SlotTapeId          slottapeid,
    @SlotUse              status,
    @SlotCurrentDrive    smallcount,

```

```

@ElementNo          elementno,
@SlotCapacity       capacity,
@SlotCurrentStatus  status
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    -- Error Checking to see if StackerId & SlotNumber exist
    IF NOT EXISTS (SELECT 1 FROM DsStSlot
                   WHERE StackerId = @StackerId AND SlotNumber = @SlotNumber)

        BEGIN
            SELECT @mystatus = 1
            raiserror 32406 "DsStSLUpdate: StackerId [%1!] and SlotNumber [%2!] do not
exist.", @StackerId, @SlotNumber

            RETURN(@mystatus)
        END

        BEGIN TRANSACTION
        UPDATE DsStSlot
        SET   StackerId      = @StackerId,
              SlotNumber     = @SlotNumber,
              SlotStatus     = @SlotStatus,
              SlotTapeId    = @SlotTapeId,
              SlotUse        = @SlotUse,
              SlotCurrentDrive = @SlotCurrentDrive,
              ElementNo      = @ElementNo,
              SlotCapacity    = @SlotCapacity,
              SlotCurrentStatus = @SlotCurrentStatus
        WHERE StackerId = @StackerId AND SlotNumber = @SlotNumber

        SELECT @mystatus = @@error

        IF (@mystatus != 0)
        BEGIN
            SELECT @mystatus = 1
            ROLLBACK TRANSACTION
            raiserror 32407 "DsStSLUpdate: Unable to update information about StackerId
[%1!] and SlotNumber [%2!].", @StackerId, @SlotNumber
        END
        ELSE
        BEGIN
            COMMIT TRANSACTION
            RETURN(@mystatus)
        END
    END
go

```

## Procedure: DsStSSelect

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStSSelect
--
-- DESCRIPTION: To get the details from DsStDevice table depending on
--               (device) Server Type for each of the various RequestIds
--               in DsStSchedule table.
--
-- TABLES ACCESSED:    DsStSchedule
--                     DsStDevice
--                     DsStConfigParameter
--
-- RETURNS:           Status (Success = 0)
--
-- INPUTS:            PARAMETER          DOMAIN
--                    -----
--                    ServerType        servertype
--
-- OUTPUTS:           PARAMETER          DOMAIN
--                    -----
--                    RequestId       requestid
--                    ServerType      servertype
--                    DeviceName     devicename
--                    Status          status
--                    OperationStatus status
--                    CurrentStatus   status
--                    CurrentOperation status
--                    CurrentVolume   drivetapeid
--
-- ****
CREATE PROCEDURE DsStSSelect
    @ServerType        servertype
AS
BEGIN
    DECLARE @status      int
    SELECT @status = 0
    SELECT "RequestId",      s.RequestId,
           "ServerType",      cp.ServerType,
           "DeviceName",      d.DeviceName,
           "Status",          d.Status,
           "OperationStatus", d.OperationStatus,
           "CurrentStatus",   d.CurrentStatus,
           "CurrentOperation",d.CurrentOperation,
           "CurrentVolume",   d.DriveTapeId
```

```

FROM   DsStSchedule s, DsStDevice d, DsStConfigParameter cp
WHERE  s.DeviceName = d.DeviceName
      AND d.ServerId = cp.ServerId
      AND cp.ServerType = @ServerType
IF (@@rowcount = 0)
BEGIN
    RAISERROR 31007 "DsStSSelect: No rows found for DeviceName "
    SELECT @status = 1
END

RETURN(@status)
END
go

```

## Procedure: DsStSSelectById

### Code

```

-- *****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStSSelectById
--
-- DESCRIPTION:          To get all the details from the DsStSchedule table
--                      for a given ScheduleId
--
-- TABLES ACCESSED:     DsStSchedule
--
-- RETURNS:              Status (Success = 0)
--                      Returns row in DsStSchedule table
--                      for the given ScheduleId
--
-- INPUTS:               PARAMETER      DOMAIN
--                      -----  -----
--                      ScheduleId    id
--
-- OUTPUTS:              PARAMETER      DOMAIN
--                      -----  -----
--                      ServerHandle   serverhandle
--
-- *****
CREATE PROCEDURE DsStSSelectById
    @ScheduleId      id
AS
BEGIN

    DECLARE @status int

    SELECT @status = 0

    IF (@ScheduleId = 0)
    BEGIN

```

```

        SELECT @status = 1
        RAISERROR 31008 "DsStSSelectById: A ScheduleId must be provided."
    END
    ELSE
    BEGIN

        SELECT "ServerHandle", ServerHandle
        FROM DsStSchedule
        WHERE ScheduleId = @ScheduleId

        IF (@@rowcount = 0)
        BEGIN
            SELECT @status = 1
            RAISERROR 31009 "DsStSSelectById: ScheduleId [%1!] is not entered in
DsStSchedule.", @ScheduleId
        END
    END
    RETURN (@status)
END
go

```

## Procedure: DsStSSelectByType

### Code

```

-- *****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStSSelectByType
--
-- DESCRIPTION:          Selects rows from the DsStSchedule Table based
--                      on Device type. Gives details of requests
--                      both in progress and queued for a given type.
--
-- TABLES ACCESSED:     DsStSchedule
--                      DsStDevice
--                      DsStConfigParameter
--
-- RETURNS:              Status (Success = 0)
--
-- INPUTS                PARAMETER                  DOMAIN
-- -----  -----
--                   ServerType                 servertype
--
-- OUTPUTS               PARAMETER                  DOMAIN
-- -----  -----
--                   ScheduleId                id
--                   Status                    status

```

```

-- RequestId           requestid
-- DeviceName          devicename
-- EstimatedDuration   duration
-- StartDate           datetime
-- EndDate              datetime
-- EstimatedStartDate  datetime
-- EstimatedEndDate    datetime
-- Priority             priority
-- ServerHandle         path
-- ****
CREATE PROCEDURE DsStSSelectByType
    @ServerType      servertype
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

-- Error checking to see if (device) ServerType requested actually exists

    IF NOT EXISTS(SELECT 1 FROM DsStConfigParameter
                  WHERE ServerType = @ServerType)
        BEGIN
            SELECT @mystatus = 1
            raiserror 31014 "DsStSSelectByType: Device ServerType [%1!] does not exist.",  

@ServerType
            RETURN (@mystatus)
        END
        SELECT      "ScheduleId",           s.ScheduleId,
                    "Status",            s.Status,
                    "RequestId",         s.RequestId,
                    "DeviceName",        s.DeviceName,
                    "EstimatedDuration", s.EstimatedDuration,
                    "StartDate",         s.StartDate,
                    "EndDate",           s.EndDate,
                    "EstimatedStartDate", s.EstimatedStartDate,
                    "EstimatedEndDate",  s.EstimatedEndDate,
                    "Priority",          s.Priority,
                    "ServerHandle",       s.ServerHandle
        FROM DsStSchedule s, DsStDevice d, DsStConfigParameter cp
        WHERE      s.DeviceName = d.DeviceName
        AND      d.ServerId = cp.ServerId
        AND      cp.ServerType = @ServerType

        SELECT @mystatus = @@error
        IF (@mystatus !=0)
            BEGIN
                SELECT @mystatus = 1
                raiserror 31015 "DsStSSelectByType: Unable to select information about
Device ServerType [%1!]", @ServerType
                RETURN (@mystatus)
            END

```

```
        RETURN (@mystatus)
END
go
```

## Procedure: DsStSSelectDeviceName

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStSSelectDeviceName
--
-- DESCRIPTION: To get the DeviceName from the DsStSchedule table
--               for a given ScheduleId
--               Called from DsStDeallocate stored procedure.
--
-- TABLES ACCESSED:   DsStSchedule
--
-- RETURNS:          Status (Success = 0)
--                   Returns row in DsStSchedule table
--                   for the given ScheduleId
--
-- INPUTS:           PARAMETER      DOMAIN
--                   -----      -----
--                   ScheduleId     id
--
-- OUTPUTS:          PARAMETER      DOMAIN
--                   -----      -----
--                   DeviceName     devicename
--
-- ****
CREATE PROCEDURE DsStSSelectDeviceName
    @ScheduleId      id,
    @DeviceName      devicename = NULL output
AS
BEGIN

    DECLARE @status int
    DECLARE @rows int

    SELECT @status = 0

    IF (@ScheduleId = 0)
    BEGIN
        SELECT @status = 1
        RAISERROR 31010 "DsStSSelectDeviceName: A ScheduleId must be provided."
    END
    ELSE
    BEGIN

        SELECT @DeviceName = DeviceName
```

```

FROM DsStSchedule
WHERE ScheduleId = @ScheduleId

SELECT @rows = @@rowcount, @status = @@error
IF (@status != 0)
BEGIN
    SELECT @status = 1
    RAISERROR 31012 "DsStSSelectDeviceName: Unable to select DeviceName for
ScheduleId [%1!] .",
                @ScheduleId
END

IF ( (@rows = 0) AND (@status=0) )
BEGIN
    SELECT @status = 1
    RAISERROR 31013 "DsStSSelectDeviceName: ScheduleId [%1!] is not entered in
DsStSchedule table.",,
                @ScheduleId
END
END
RETURN (@status)

END
go

```

## Procedure: DsStSTDelete

### Code

```

-- *****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStSTDelete
--
-- DESCRIPTION:      Delete a row about Server Types.
--
-- TABLES ACCESSED:  DsStServerType
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER      DOMAIN
--                   -----
--                   ServerType     servertype
--
-- OUTPUTS:          PARAMETER      DOMAIN
--                   -----
--                   NONE
-- *****
CREATE PROCEDURE DsStSTDelete
    @ServerType      servertype = null

```

```

AS
BEGIN
    DECLARE @mystatus int

    SELECT @mystatus = 0

    -- Error checking to see if trying to delete a valid ServerType
    IF NOT EXISTS (SELECT 1 FROM DsStServerType WHERE ServerType = @ServerType)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32500 "DsStSTDelete: ServerType [%1!] does not exist.", @ServerType
        RETURN @mystatus
    END

    -- Error checking to see if said ServerType is present in the
    -- DsStConfigParameter Table.
    -- If ServerType is currently being used in DsStConfigParameter Table
    -- then ServerType is not eligible for deletion.
    IF EXISTS (SELECT 1 FROM DsStConfigParameter where ServerType = @ServerType)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32501 "DsStSTDelete: ServerType [%1!] is unable to be deleted because it is
currently being used in the DsStConfigParameter table.", @ServerType
        RETURN (@mystatus)
    END

    -- Delete the ServerType
    BEGIN TRANSACTION
    DELETE DsStServerType
    WHERE ServerType = @ServerType

    SELECT @mystatus = @@error

    IF (@mystatus != 0)
    BEGIN
        ROLLBACK TRANSACTION
        SELECT @mystatus = 1
        raiserror 32502 "DsStSTDelete: Unable to delete ServerType [%1!].", @ServerType
        RETURN (@mystatus)
    END

    COMMIT TRANSACTION
    RETURN (@mystatus)
END
go

```

## Procedure: DsStSTInsert

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStSTInsert
--
-- DESCRIPTION:      Inserts a row into the DsStServerType table.
-- CREATE DATE:      June 12, 1997
-- DEVELOPER:        Lisa Colombo
--
-- DATE MODIFIED:10/24/97; LJC - modified columns
--
-- TABLES ACCESSED:  DsStServerType
--
-- RETURNS:          Status (Success = 0)
--
-- INPUT PARAMETERS      DOMAIN
-- -----
-- ServerType            servertype
-- ServerDescription     description
-- PollingRequired       flag
--
-- OUTPUT PARAMETERS      DOMAIN
-- -----
-- NONE
-- ****
CREATE PROCEDURE DsStSTInsert
    @ServerType      servertype = null,
    @ServerDescription  description = null,
    @PollingRequired   flag = null
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0
    -- Error checking to see if ServerType is blank or null
    IF (@ServerType is null or @ServerType = "")
        BEGIN
            SELECT @mystatus = 1
            raiserror 32503 "DsStSTInsert: Server Type may not be blank or null."
            RETURN (@mystatus)
        END
    -- Error checking to see if ServerType already exists
    IF EXISTS (Select 1 from DsStServerType where ServerType = @ServerType)
        BEGIN
            SELECT @mystatus = 1
        END
```

```

raiserror 32504 "DsStSTInsert: Unable to insert Server Type [%1!] as server already
exists.", @ServerType
      RETURN @mystatus
END

BEGIN TRANSACTION
INSERT DsStServerType
  (ServerType,
   ServerDescription,
   PollingRequired)
VALUES (@ServerType,
        @ServerDescription,
        @PollingRequired)

SELECT @mystatus = @@error

IF (@mystatus != 0)
BEGIN
  SELECT @mystatus = 1
  ROLLBACK TRANSACTION
  raiserror 32505 "DsStSTInsert: Unable to insert ServerType [%1!].", @ServerType
END

COMMIT TRANSACTION
RETURN (@mystatus)
END
go

```

## Procedure: DsStSTSelect

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStSTSelect
--
-- DESCRIPTION:      Selects list of Server Types except Stg Disk.
-- DATE CREATED: 3/25/97
--
-- DATE MODIFIED:8/27/97
-- MODIFIED BY:      Lisa Colombo
-- MODIFICATION: Staging Disk not excluded from list when no. of
--                  DsStConfigParameter STAGING MONITORs = 0 &
--                  nothing returned for Server Type List when no
--                  records in DsStConfigParameter table;
--                  Table join eliminated from 2nd statement of union.
--
-- TABLES ACCESSED:  DsStServerType
--
-- RETURNS:          Status (Success = 0)
-- 
```

```

-- INPUTS:  PARAMETER      DOMAIN
----- -----
--      NONE

-- OUTPUTS: PARAMETER      DOMAIN
----- -----
--      NumberOfServers    int
--      ServerType         servertype
--      ServerDescription  description
--

-- ****
CREATE PROCEDURE DsStSTSelect
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

    -- Error checking to see if any Server Types exist.
    -- In the current system this is really an error due to
    -- the fact that no functionality currently exists to create
    -- new Server Types.

    IF NOT EXISTS(select * from DsStServerType)
    BEGIN
        SELECT @mystatus = 1
        raiserror 32506 "DsStServerType: No Server Types exist."
        RETURN (@mystatus)
    END

    -- Select to fill GUI Screen "Server Type Information"
    -- Requires Server Type, # of Servers and Description
    -- STAGING DISK is excluded for the reason that when a STAGING MONITOR
    -- Server is configured a STAGING DISK server will be configured
    -- behind the scenes.

    SELECT  "NumberOfServers",  count(*),
            "ServerType",      st.ServerType,
            "ServerDescription", ServerDescription
    FROM  DsStServerType st, DsStConfigParameter cp
    WHERE st.ServerType != "STAGING DISK"
    AND st.ServerType = cp.ServerType
    GROUP BY st.ServerType

    UNION

    SELECT  "NumberOfServers",  0,
            "ServerType",      st.ServerType,
            "ServerDescription", ServerDescription
    FROM  DsStServerType st
    WHERE st.ServerType != "STAGING DISK"
    AND st.ServerType NOT IN
        (SELECT ServerType FROM DsStConfigParameter)

```

```

ORDER BY st.ServerType

SELECT @mystatus = @@error
IF (@mystatus != 0)
BEGIN
    SELECT @mystatus = 1
    raiserror 32507 "DsStSTSelect: Unable to select Server Type information."
    RETURN (@mystatus)
END

RETURN (@mystatus)
END
go

```

## Procedure: DsStSTUpdate

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStSTUpdate
--
-- DESCRIPTION:      Update a row about Server Types.
-- CREATE DATE:     03/25/97
-- CREATED BY:      mgowans
--
-- DATE MODIFIED:   10/24/97; LJC - modified columns
--
-- TABLES ACCESSED: DsStServerType
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER          DOMAIN
-- -----           -----
--       ServerType      servertype
--       ServerDescription  description
--       PollingRequired   flag
--
-- OUTPUTS:          PARAMETER          DOMAIN
-- -----           -----
--       NONE
-- ****

CREATE PROCEDURE DsStSTUpdate
    @ServerType      servertype = null,
    @ServerDescription  description = null,
    @PollingRequired   flag = null
AS
BEGIN
    DECLARE @mystatus int

    SELECT @mystatus = 0

```

```

-- Error checking to see if ServerType is valid
IF NOT EXISTS (SELECT 1 FROM DsStServerType WHERE ServerType = @ServerType)
BEGIN
    SELECT @mystatus = 1
    raiserror 32508 "DsStSTUpdate: Server Type [%1!] does not exist.", @ServerType
    RETURN (@mystatus)
END

BEGIN TRANSACTION
UPDATE DsStServerType
    SET ServerDescription = @ServerDescription,
        PollingRequired = @PollingRequired
    WHERE ServerType = @ServerType

SELECT @mystatus = @@error

IF (@mystatus != 0)
BEGIN
    ROLLBACK TRANSACTION
    SELECT @mystatus = 1
    raiserror 32509 "DsStSTUpdate: Unable to update Server Type [%1!].", @ServerType
    RETURN (@mystatus)
END

COMMIT TRANSACTION
RETURN (@mystatus)
END
go

```

## Procedure: DsStSUpdate

### Code

```

-- ****
-- BEGIN PROLOG
-- 
-- PROCEDURE NAME:      DsStSUpdate
-- 
-- DESCRIPTION:          Updates the Status with "inprogress" in the
--                      DsStSchedule table for a ScheduleId.
--                      Updates the DeviceName with the given DeviceName .
--                      Also updates the ServerHandle with the given ServerHandle.
--                      Called from DsStScheduler stored procedure.
-- 
-- TABLES ACCESSED:     DsStSchedule
-- 
-- RETURNS:              Status (Success = 0)
-- 
-- INPUTS:      PARAMETER          DOMAIN
--             -----      -----
--             ScheduleId      id

```

```

--      DeviceName      devicename
--      ServerHandle    path
--
-- OUTPUTS:  PARAMETER      DOMAIN
----- ----- -----
--      NONE
-- ****
CREATE PROCEDURE DsStSUpdate
    @ScheduleId   id,
    @DeviceName    devicename,
    @ServerHandle  path
AS
BEGIN

DECLARE @status int
DECLARE @rows int
DECLARE @EstEndDate datetime

SELECT @status = 0
SELECT @rows = 0

IF (@ScheduleId = null )
BEGIN
    SELECT @status = 1
    RAISERROR 31051 "DsStSUpdate: A ScheduleId must be provided."
END
ELSE

IF (@DeviceName = null )
BEGIN
    SELECT @status = 1
    RAISERROR 31052 "DsStSUpdate: A DeviceName must be provided."
END
ELSE
IF (@ServerHandle = null)
BEGIN
    SELECT @status = 1
    RAISERROR 31053 "DsStSUpdate: A ServerHandle must be provided."
END
ELSE
BEGIN
/*
SELECT @EstEndDate = dateadd(hour, EstimatedDuration, StartDate)
FROM  DsStSchedule
WHERE ScheduleId = @ScheduleId
*/
BEGIN TRANSACTION
UPDATE DsStSchedule
SET DeviceName      = @DeviceName,
    ServerHandle    = @ServerHandle,
    Status          = 1

```

```

--      StartDate      = getdate(),
--      EstimatedStartDate = getdate(),
--      EndDate      = getdate(),
--      EstimatedEndDate   = getdate()
--      EstimatedEndDate = @EstEndDate
--      EstimatedEndDate = dateadd(hour, EstimatedDuration, StartDate)
WHERE ScheduleId      = @ScheduleId

SELECT @rows = @@rowcount
SELECT @status = @@error

IF (@status != 0)
BEGIN
    ROLLBACK TRANSACTION
    RAISERROR 31054 "DsStSUpdate: Update Failed for [%1!] in DsStSchedule table.",
                    @ScheduleId
    RETURN(@status)
END

IF (@rows = 0)
BEGIN
    SELECT @status = 1
    ROLLBACK TRANSACTION
    RAISERROR 31055 "DsStSUpdate: ScheduleId [%1!] not in DsStSchedule
table.",@ScheduleId
    RETURN(@status)
END

COMMIT TRANSACTION
END -- for BEGIN ELSE
RETURN(@status)
END
go

```

## **Procedure: DsStSUpdateStatus**

### **Code**

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsStSUpdateStatus
--
-- DESCRIPTION: Updates in the DsStSchedule table
--               the Status with the given status "complete"
--
-- TABLES ACCESSED:    DsStSchedule
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER      DOMAIN
-- -----  -----  -----

```

```

--      ScheduleId      id
--      Status          status
--
-- OUTPUTS:  PARAMETER      DOMAIN
-----  

-- ****
CREATE PROCEDURE DsStSUpdateStatus
    @ScheduleId  id,
    @Status       status
AS
BEGIN

    DECLARE @status int
    DECLARE @rows int

    SELECT @status = 0
    SELECT @rows = 0

    IF @ScheduleId = 0
    BEGIN
        SELECT @status = 1
        RAISERROR 31081 "DsStSUpdateStatus: A ScheduleId must be provided."
    END
    ELSE
    IF @Status > 2
    BEGIN
        SELECT @status = 1
        RAISERROR 31082 "DsStSUpdateStatus: Invalid Status."
    END
    ELSE

    BEGIN
        BEGIN TRANSACTION

        UPDATE DsStSchedule
        SET Status = @Status
        WHERE ScheduleId = @ScheduleId

        SELECT @rows = @@rowcount, @status = @@error

        IF (@status != 0)
        BEGIN
            SELECT @status = 1
            ROLLBACK TRANSACTION
            RAISERROR 31083 "DsStSUpdateStatus: Update Failed for [%1!] in DsStSchedule table
",
            @ScheduleId
        END
        IF ((@rows = 0) AND (@status != 0) )
        BEGIN

```

```

select @status = 1
ROLLBACK TRANSACTION
RAISERROR 31084 "DsStSUpdateStatus: ScheduleId [%1!] not in DsStSchedule table",
@ScheduleId
END

```

```

COMMIT TRANSACTION
END
RETURN(@status)
END
go

```

## Procedure: DsStVGInsert

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStVGInsert
--
-- DESCRIPTION:      Insert a record into the DsStVolumeGroup Table.
--                   This procedure will accommodate Volume Groups
--                   being inserted for the very first time. When
--                   the path to a current volume group is changed
--                   the DsStVGUpdate procedure will be called.
--
-- TABLES ACCESSED:  DsStVolumeGroup
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER          DOMAIN
--                   -----
--                   ServerId        serverid
--                   VolumeGroupName volumegroupname
--                   VolumeGroupPath path
--
-- OUTPUTS:          PARAMETER          DOMAIN
--                   -----
--                   NONE
-- ****
CREATE PROCEDURE DsStVGInsert
    @ServerId        serverid = null,
    @VolumeGroupName volumegroupname = null,
    @VolumeGroupPath path = null
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0

```

```

-- Error Checking to see if ServerId exists in DsStConfigParameter
-- Table
IF NOT EXISTS (SELECT 1 FROM DsStConfigParameter WHERE
    ServerId = @ServerId)
BEGIN
    SELECT @mystatus = 1
    raiserror 35004 "DsStVGInsert: The ServerId you are attempting to insert [%1!] does not
currently have an entry in the DsStConfigParameter Table.", @ServerId
    RETURN (@mystatus)
END

-- Error checking to see if Volume Group Name is blank or null
IF (@VolumeGroupName = "" or @VolumeGroupName is null)
BEGIN
    SELECT @mystatus = 1
    raiserror 35005 "DsStVGInsert: The Volume Group Name may not be blank or null."
    RETURN (@mystatus)
END

-- Error checking to see if Volume Group Path is blank or null
IF (@VolumeGroupPath = "" or @VolumeGroupPath is null)
BEGIN
    SELECT @mystatus = 1
    raiserror 35006 "DsStVGInsert: The Volume Group Path may not be blank or null."
    RETURN (@mystatus)
END

-- Error checking to see if Volume Group Name already exists for ServerId
IF EXISTS (SELECT 1 from DsStVolumeGroup where ServerId = @ServerId AND
    VolumeGroupName = @VolumeGroupName)
BEGIN
    SELECT @mystatus = 1
    raiserror 35007 "DsStVGInsert: Unable to Insert Volume Group Name [%1!] for ServerId
[%2!] as this Volume Group already exists.", @ServerId, @VolumeGroupName
    RETURN @mystatus
END

BEGIN TRANSACTION
INSERT DsStVolumeGroup (ServerId,
    VolumeGroupName,
    VolumeGroupPath,
    VolumeStartDate,
    VolumeEndDate)
values
    (@ServerId,
    @VolumeGroupName,
    @VolumeGroupPath,
    getdate(),
    NULL)

SELECT @mystatus = @@error

```

```

IF (@mystatus != 0)
BEGIN
    ROLLBACK TRANSACTION
    SELECT @mystatus = 1
    raiserror 35008 "DsStVGInsert: Unable to insert ServerId [%1!], Volume Group Name
    [%2!], Volume Group Path [%3!].", @ServerId, @VolumeGroupName, @VolumeGroupPath
    RETURN @mystatus
END
COMMIT TRANSACTION
RETURN @mystatus
END
go

```

## Procedure: DsStVGSelectById

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:   DsStVGSelectById
--
-- DESCRIPTION:      Select all current volume groups and associated
--                   paths by ServerId.
--
-- TABLES ACCESSED:  DsStVolumeGroup
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER          DOMAIN
--                   -----
--                   ServerId        serverid
--
-- OUTPUTS:          PARAMETER          DOMAIN
--                   -----
--                   ServerId        serverid
--                   VolumeGroupName  volumegroupname
--                   VolumeGroupPath  path
--                   VolumeStartDate  datetime
--                   VolumeEndDate    datetime
--
-- ****
CREATE PROCEDURE DsStVGSelectById
    @ServerId          serverid = null
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0
    -- Error checking to see if valid ServerId
    IF NOT EXISTS (SELECT 1 FROM DsStConfigParameter WHERE

```

```

        ServerId = @ServerId and ServerType = "ARCHIVE")
BEGIN
    SELECT @mystatus = 1
    raiserror 35000 "DsSTVGSelectById: ServerId [%1!] with a Server Type of ARCHIVE
does not exist.", @ServerId
    RETURN (@mystatus)
END

SELECT "ServerId",           ServerId,
      "VolumeGroupName",   VolumeGroupName,
      "VolumeGroupPath",   VolumeGroupPath,
      "VolumeStartDate",   VolumeStartDate,
      "VolumeEndDate",     VolumeEndDate
FROM DsStVolumeGroup
WHERE ServerId      = @ServerId
  AND VolumeEndDate IS NULL

SELECT @mystatus = @@error
IF (@mystatus != 0)
BEGIN
    SELECT @mystatus = 1
    raiserror 35001 "DsStVGSelectById: Unable to select Volume Group information about
ServerId [%1!].", @ServerId
    RETURN (@mystatus)
END

RETURN (@mystatus)
END
go

```

## Procedure: DsStVGSelectHistory

### Code

```

-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStVGSelectHistoryById
--
-- DESCRIPTION:      Select all volume groups associated with a
--                   ServerId and VolumeGroupName.
--
-- TABLES ACCESSED:  DsStVolumeGroup
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER      DOMAIN
--                   -----
--                   ServerId       serverid
--                   VolumeGroupName volumegroupname
--
-- OUTPUTS:          PARAMETER      DOMAIN

```

```

-- ----- -----
--      ServerId      serverid
--      VolumeGroupName    volumegroupname
--      VolumeGroupPath    path
--      VolumeStartDate    datetime
--      VolumeEndDate    datetime
-- *****

CREATE PROCEDURE DsStVGSelectHistory
    @ServerId      serverid = null,
    @VolumeGroupName    volumegroupname = null
AS
BEGIN
    DECLARE @mystatus int

    SELECT @mystatus = 0

    -- Error checking to see if History available for
    -- ServerId, VolumeGroupName
    IF NOT EXISTS (SELECT 1 FROM DsStVolumeGroup WHERE
        ServerId = @ServerId AND VolumeGroupName =
        @VolumeGroupName)
        BEGIN
            SELECT @mystatus = 1
            raiserror 35002 "DsStVGSelectHistory: ServerId [%1!], Volume Group Name [%2!] does
not exist.", @ServerId, @VolumeGroupName
            RETURN (@mystatus)
        END

    SELECT "ServerId",      ServerId,
           "VolumeGroupName",    VolumeGroupName,
           "VolumeGroupPath",    VolumeGroupPath,
           "VolumeStartDate",    VolumeStartDate,
           "VolumeEndDate",    VolumeEndDate
    FROM DsStVolumeGroup
    WHERE ServerId      = @ServerId
        AND VolumeGroupName    = @VolumeGroupName
    ORDER BY VolumeStartDate desc

    SELECT @mystatus = @@error
    IF (@mystatus != 0)
        BEGIN
            SELECT @mystatus = 1
            raiserror 35003 "DsStVGSelectHistory: Unable to select history for ServerId [%1!], with a
Volume Group Name of [%2!].", @ServerId, @VolumeGroupName

            RETURN (@mystatus)
        END

    RETURN (@mystatus)
END
go

```

## Procedure: DsStVGUpdate

### Code

```
-- ****
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    DsStVGUpdate
--
-- DESCRIPTION:      Update a record into the DsStVolumeGroup Table.
--
-- TABLES ACCESSED:  DsStVolumeGroup
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:           PARAMETER      DOMAIN
-- ----- -----
--     ServerId       serverid
--     VolumeGroupName volumegroupname
--     CurrentVolumeGroupPath path
--     NewVolumeGroupPath   path
--
-- OUTPUTS:          PARAMETER      DOMAIN
-- ----- -----
--     NONE
-- ****
CREATE PROCEDURE DsStVGUpdate
    @ServerId       serverid = null,
    @VolumeGroupName volumegroupname = null,
    @CurrentVolumeGroupPath path = null,
    @NewVolumeGroupPath   path = null
AS
BEGIN
    DECLARE @mystatus int
    SELECT @mystatus = 0
    -- Error Checking to see if CurrentVolumeGroupPath and
    -- NewVolumeGroupPath are the same
    IF (@CurrentVolumeGroupPath = @NewVolumeGroupPath)
        BEGIN
            SELECT @mystatus = 1
            raiserror 35009 "DsStVGUpdate: The Current Volume Group Path [%1!] and the New
Volume Group Path [%2!] may not be the same.", @CurrentVolumeGroupPath,
@NewVolumeGroupPath
            RETURN (@mystatus)
        END
    -- Error checking to see if NewVolumeGroupPath is blank or null
    IF (@NewVolumeGroupPath = null or @NewVolumeGroupPath = "")
        BEGIN
            SELECT @mystatus = 1
```

```

raiserror 35010 "DsStVGUpdate: The New Volume Group Path may not be blank or null."
    RETURN (@mystatus)
END

-- Error Checking to see if row that is being updated
-- actually exists.
IF NOT EXISTS (SELECT 1 FROM DsStVolumeGroup WHERE
    ServerId = @ServerId AND
    VolumeGroupPath = @CurrentVolumeGroupPath AND
    VolumeGroupName = @VolumeGroupName AND
    VolumeEndDate IS NULL)
BEGIN
    SELECT @mystatus = 1
    raiserror 35011 "DsStVGUpdate: ServerId [%1!], Volume Group Name [%2!], Current
Volume Group Path [%3!] may not be closed out because it does not currently exist.",
    @ServerId, @VolumeGroupName, @CurrentVolumeGroupPath
        RETURN (@mystatus)
END

BEGIN TRANSACTION
UPDATE DsStVolumeGroup
    SET VolumeEndDate = getdate()
WHERE ServerId = @ServerId
    AND VolumeGroupName = @VolumeGroupName
    AND VolumeGroupPath = @CurrentVolumeGroupPath
    AND VolumeEndDate IS NULL

SELECT @mystatus = @@error
IF (@mystatus != 0)
BEGIN
    ROLLBACK TRANSACTION
    SELECT @mystatus = 1
    raiserror 35012 "DsStVGUpdate: The Volume Group [%1!] attached to ServerId [%2!] that
you attempted to update failed.", @VolumeGroupName, @ServerId
        RETURN (@mystatus)
END

INSERT DsStVolumeGroup (ServerId,
    VolumeGroupName,
    VolumeGroupPath,
    VolumeStartDate,
    VolumeEndDate)
values (@ServerId,
    @VolumeGroupName,
    @NewVolumeGroupPath,
    getdate(),
    NULL)
SELECT @mystatus = @@error

IF (@mystatus != 0)
BEGIN
    ROLLBACK TRANSACTION

```

```

SELECT @mystatus = 1
raiserror 35013 "DsStVGUpdate: ServerId [%1!], Volume Group Name [%2!], Volume
Group Path [%3!] was unable to be inserted.", @ServerId, @VolumeGroupName,
@NewVolumeGroupPath
RETURN @mystatus
END

COMMIT TRANSACTION
RETURN @mystatus
END
go

```

## **4.2 File Usage**

There are cases when the implementation of a persistent data requirement is better suited to a flat file than to a database table. A typical example of such data is system configuration information. System configuration information is fairly static and usually has no explicit relationship to other data in the enterprise. Another common use of files in ECS is as an interface mechanism between ECS and the external world. FIILSUB file usage is detailed in this section via file definitions, attribute definitions, and attribute domain definitions.

### **4.2.1 Files Definitions**

A listing of each the files in the STMGMT database is given here. A brief definition of each of these files follows.

To Be Supplied (TBS)

### **4.2.2 Attributes**

Brief definitions of each of the attributes present in the files defined above are contained herein.  
TBS

### **4.2.3 Attribute Domains**

Domains represent the ranges of valid values allowed for a given file attribute. Attributes domains for each of the attributes defined above are given here.

TBS

## **5. Performance and Tuning Factors**

---

### **5.1 Indexes**

An index provides a means of locating a row in a table based on the value of specific columns, without having to scan each row in the table. If used appropriately, indexes can significantly increase data retrieval. Sybase allows the definition of two types of indexes, clustered and non-clustered. In a clustered index, the rows in a table are physically stored in the sort order determined by the index. Clustered indexes are particularly useful, when the data is frequently retrieved in order. Non-clustered indexes differ from their clustered counterpart, in that data is not physically stored in sort order. Only one clustered index may be defined per table. A list of all the indexes defined against tables in the STMGMT database is given here along with a description of each index..

### **5.2 Segments**

Sybase supports the definition of segments. A segment is a named pointer to a storage device or devices. Segments are used to manually place database objects onto particular storage devices. All segments explicitly defined in the STMGMT database are described herein.

### **5.3 Named Caches**

A cache is a block of memory that is used by Sybase to house data pages that are currently being accessed. A named cache is a named block of memory that the SQL server can use to house frequently accessed tables. Assigning a table to cache causes it to be loaded into memory. This greatly increases performance by eliminating the time expense normally associated with disk i/o. Named caches used in the STMGMT databases are described herein.

This page intentionally left blank.

## 6. Database Security

---

### 6.1 Initial Users

Upon initial installation the following users will have access to STMGMT database. The level of access is limited to that associated with their assigned group and/or role. A complete definition of each of these groups and roles is given below.

TBS

### 6.2 Groups

Groups are a means of logically associating users with similar data access needs. Once a group has been defined, object and command permissions can be granted to that group. A user who is member of a group inherits all of the permissions granted to that group. Several group have been defined in the STMGMT database upon initial installation. A definition of each of these groups is contained herein.

TBS

### 6.3 Roles

Roles were introduced in Sybase 10 to allow a structured means for granting users the permissions needed to perform standard database administration activities and also provide a means for easily identifying such users. There are six pre-defined roles that may be assigned to a user. A definition of each of these roles follows as well as a description of the types of activities that may be performed by each role.

**System Administrator** (sa\_role) - This role is used to grant a specific user to permissions needed to perform standard system administrator duties including:

- a. installing SQL server and specific SQL server modules
- b. managing the allocation of physical storage
- c. tuning configuration parameters
- d. creating databases

**Site Security Officer** (sso\_role) - This role is used to grant a specific user the permissions needed to maintain SQL server security including:

- a. adding server logins
- b. administrating passwords

- c. managing the audit system
- d. granting users all roles except sa\_role

**Operator** (oper\_role) - This role is used to grant a specific user the permissions needed to manage backup and recovery of the database including;

- a. dumping transactions and databases
- b. loading transactions and databases

**Navigator** (navigator\_role) -This role is used to grant a specific user the permissions needed to manage the navigation server.

**Replication** (replication\_role) - - This role is used to grant a specific user the permissions needed to manage the replication server.

**Sybase Technical Support** (sybase\_ts\_role) - This role is used to grant a specific user the permissions needed to perform database consistency checker (dbcc), a sybase supplied utility, commands that are considered outside of the realm of normal system administrator activities.

## 6.4 Object Permissions

TBS

# **7. Replication**

---

## **7.1 Replication Overview**

Replication as the name implies is a set of Sybase products that allow replication of data from one database to another. In order for replication to be accomplished the data source must define the tables and columns that may be replicated to a data recipient. These definitions are referred to replication definitions. In the same manner a data recipient must specify the replication definitions in which he is interested. These specifications are referred to as replication subscriptions. In addition the replication database and server must be configured to support the potentially large volumes of data that will be transferred between the source and recipient databases. Storage Management does not utilize replication.

## **7.2 Replication Definitions**

Not applicable

## **7.3 Replication Subscriptions**

Not applicable

## **7.4 Replication Database Configuration**

Not applicable

## **7.5 Replication Server Configuration**

Not applicable

This page intentionally left blank.

## **8. Scripts**

---

### **8.1 Installation Scripts**

Any scripts used to support installation of the STMGMT database are described herein.. These files may be found in the directory /ecs/formal/DSS/stmgmt/src/database.

<b>Script File</b>	<b>Description</b>
DsStBuildAll	Installs/populates Storage management database

### **8.2 De-Installation Scripts**

Any scripts used to support de-installation of the STMGMT database are described herein.

TBS

### **8.3 Backup/Recovery Scripts**

Any scripts used to facilitate backup or recovery of the STMGMT database are described herein.

TBS

### **8.4 Miscellaneous Scripts**

Miscellaneous scripts applicable to the STMGMT database are described herein.

TBS

This page intentionally left blank.

## Abbreviations and Acronyms

---

ACL	Access Control List
ACMHW	Access and Control Management HWCI
ADC	affiliated data center
ADSHW	Advertising Server HWCI
ADSRV	Advertising Service CSCI
AI&T	algorithm integration and test
AITHW	Algorithm Integration and Test HWCI
AITTL	Algorithm Integration and Test CSCI
AM-1	EOS AM Project spacecraft 1, morning spacecraft series -- ASTER, CERES, MISR, MODIS and MOPITT instruments
ANSI	American National Standards Institute
API	application program (or programming) interface
APID	application's process ID
AQAHW	Algorithm QA HWCI
ASCII	American Standard Code for Information Exchange
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer (formerly ITIR)
AVHRR	Advanced Very High-Resolution Radiometer
BER	bit error rate
BUFR	binary universal format for representation of data
CASE	Computer Aided Software Engineering
CCSDS	Consultative Committee for Space Data Systems
CD	contractual delivery 214-001
CD-ROM	compact disk -- read only memory
CDR	Critical Design Review
CDRL	contract data requirements list
CERES	Clouds and Earth's Radiant Energy System

CI	configuration item
COTS	commercial off-the-shelf (hardware or software)
CPU	central processing unit
CSCI	computer software configuration item
CSDT	Computer Science Data Type
CSMS	Communications and Systems Management Segment (ECS)
CSS	Communications Subsystem
DAAC	Distributed Active Archive Center
DAN	data availability notice
DAO	Data Assimilation Office
DAR	data acquisition request
DAS	data availability schedule
DBMS	Database Management System
DDICT	Data Dictionary CSCI
DDIST	Data Distribution Services CSCI
DDSRV	Document Data Server CSCI
DESKT	Desktop CSCI
DID	data item description
DIM	distributed information manager (SDPS)
DIMGR	Distributed Information Manager CSCI
DIPHW	Distribution and Ingest Peripheral Management HWCI
DMGHW	Data Management HWCI
DMS	Data Management Subsystem
DMWG	Data Management Working Group
DP	Data Provider
DPR	data processing request
DPREP	Science Data Preprocessing CSCI
DPS	Data Processing Subsystem
DRPHW	Data Repository HWCI

DSS	Data Server Subsystem
ECS	EOSDIS Core System
EDC	EROS Data Center
EDHS	ECS Data Handling System
EDOS	EOS Data and Operations System
EOS	Earth Observing System
EOS-AM	EOS Morning Crossing (Descending) Mission -- see AM-1
EOSDIS	Earth Observing System Data and Information System
EROS	Earth Resources Observation System
ESDIS	Earth Science Data and Information System (GSFC)
ESDT	Earth science data types
ESN	EOSDIS Science Network (ECS)
FDDI	fiber distributed data interface
FDF	flight dynamics facility
FDFEPHEM	FDF-generated definitive orbit data
FGDC	Federal Geographic Data Committee
FK	Foreign Key
FOO	Flight of Opportunity
FOS	Flight Operations Segment (ECS)
GB	gigabyte ( $10^9$ )
GNU	(recursive acronym: “GNU’s Not Unix”); a project supported by the Free Software Foundation dedicated to the delivery of free software
GPCP	Global Precipitation Climatology Project
GPCP	Global Precipitation Climatology Project
GPI	GOES Precipitation Index
GRIB	GRid In Binary
GSFC	Goddard Space Flight Center
GTWAY	Version 0 Interoperability Gateway CSCI
GUI	graphic user interface

GV	ground validation
HDF	hierarchical data format
HDF-EOS	an EOS proposed standard for a specialized HDF data format
HIPPI	high performance parallel interface
HMI	human machine interface
HTML	HyperText Markup Language
HTTP	Hypertext Transport Protocol
HWCI	hardware configuration item
I&T	integration and test
I/F	interface
I/O	input/output
ICD	interface control document
ICLHW	Ingest Client HWCI
ID	identification
IDE	Interactive Development Environments
IDG	Infrastructure Development Group
IDR	Incremental Design Review
IERS	International Earth Rotation Service
IMS	Information Management System (obsolete ECS element name)
INGST	Ingest Services CSCI
IOS	Interoperability Subsystem
IP	international partners
IR-1	Interim Release 1
IRD	interface requirements document
ISO	International Standards Organization
ISS	Internetworking Subsystem
IV&V	independent verification and validation
JPL	Jet Propulsion Laboratory
L0-L4	Level 0 (zero) through Level 4

LaRC	Langley Research Center (DAAC)
LIM	local information manager (SDPS)
LIMGR	Local Information Manager CSCI
LIS	Lightning Imaging Sensor
LSM	local system management (ECS)
MB	megabyte ( $10^6$ )
MDT	mean downtime
MDT	mean downtime
MFLOPS	mega (millions of) floating-point operations ( $10^6$ ) per second
MISR	Multi-Angle Imaging SpectroRadiometer
MODIS	Moderate-Resolution Imaging Spectrometer
MOPITT	Measurements of Pollution in the Troposphere
MSFC	Marshall Space Flight Center
MSS	Management Support Subsystem
MTBF	mean time between failure
MTPE	Mission to Planet Earth
MTTR	mean time to restore
N/A	not applicable
NAS	National Academy of Science
NASA	National Aeronautics and Space Administration
NESDIS	National Environmental Satellite Data and Information Service
NMC	National Meteorological Center (NOAA)
NOAA	National Oceanic and Atmospheric Administration
NSIDC	National Snow and Ice Data Center (DAAC)
O/A	orbit/altitude
ODC	other data center
OSI	Open System Interconnect
PDPS	Planning and Data Processing Subsystem
PDR	Preliminary Design Review

PDS	production data set
PGE	Product Generation Executive
PGS	Product Generation System (obsolete ECS element name) (ASTER)
PK	Primary Key
PLANG	Production Planning CSCI
PLNHW	Planning HWCI
PLS	Planning Subsystem
POSIX	Portable Operating System Interface for Computer Environments
PR	Precipitation Radar (TRMM)
PRONG	Processing CSCI
QA	quality assurance
RMA	reliability, maintainability, availability
RTF	rich text format
SAA	satellite active archive
SAGE	Stratospheric Aerosol and Gas Experiment
SCF	Science Computing Facility
SDP	Science Data Processing
SDPF	Sensor Data Processing Facility (GSFC)
SDPS	Science Data Processing Segment (ECS)
SDPTK	SDP Toolkit CSCI
SDSRV	Science Data Server CSCI
SeaWIFS II	Sea-Viewing Wide Field-of-View Sensor II
SFDU	Standard Format Data Unit
SMC	System Management Center (ECS)
SPRHW	Science Processing HWCI
SRS	software requirements specification
SSM/I	Special Sensor for Microwave/Imaging (DMSP)
SST	sea surface temperature
STMGMT	Storage Management

STMGT	Storage Management Software CSCI
SUBSRV	Subscription Server
TMI	TRMM Microwave Image
TOMS	Total Ozone Mapping Spectrometer
TONS	TDRS On-board Navigational System
TRMM	Tropical Rainfall Measuring Mission (joint US-Japan)
TSDIS	TRMM Science Data and Information System
USNO	US Naval Observatory
UT	universal time
UTC	universal time code
V0	Version 0
VIRS	Visible Infrared Scanner (TRMM)
WAIS	Wide Area Information Server
WKBCH	Workbench CSCI
WKSHW	Working Storage HWCI
WWW	World-Wide Web

This page intentionally left blank.